OSCAT

Network:LIBRARY

Dokumentation

Version 1.21

Inhaltsverzeichnis

1. Rechtsgrundlagen

Die OSCAT Network Bibliothek definiert neben den Standard Datentypen weitere Datentypen. Diese werden innerhalb der Bibliothek verwendet, können aber jederzeit von Anwender für eigene Deklarationen verwendet werden. Ein Löschen oder verändern von Datentypen kann dazu führen das Teile der Bibliothek sich nicht mehr kompilieren lassen.

1.1. Haftungsausschluss

Die in der OSCAT Bibliothek enthaltenen Softwaremodule sind in der Absicht angeboten, als Entwicklungsvorlage und Leitfaden zur Softwareentwicklung für SPS nach IEC61131-3 zu dienen. Eine Funktionsgarantie wird von den Entwicklern nicht übernommen und wird explizit ausgeschlossen. Da die in der Bibliothek enthaltenen Softwaremodule ohne jegliche Kosten bereitgestellt werden, besteht keinerlei Gewährleistung, soweit dies gesetzlich zulässig ist. Sofern nicht explizit schriftlich vereinbart, stellen die Copyright-Inhaber und / oder Dritte die Softwaremodule so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich - aber nicht begrenzt - auf Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko und die volle Verantwortung bezüglich Qualität, Fehlerfreiheit und Leistungsfähigkeit der Softwaremodule liegen beim Anwender selbst. Sollte sich die Bibliothek, oder Teile der Bibliothek als fehlerhaft erweisen, liegen die Kosten für notwendigen Service, Reparatur und / oder Korrektur beim Anwender selbst. Sollten Teile oder die gesamte Bibliothek zur Erstellung von Anwendungssoftware verwendet werden, oder in Softwareprojekten eingesetzt werden, so haftet der Anwender für die Fehlerfreiheit, Funktion und Qualität der Anwendung. Eine Haftung durch OSCAT ist grundsätzlich ausgeschlossen.

Der Anwender der OSCAT Bibliothek hat durch geeignete Tests und Freigaben und Qualitätssicherungsmaßnahmen dafür zu sorgen, dass durch eventuelle Fehler in der Bibliothek von OSCAT keine Schäden entstehen können. Die vorstehenden Lizenzbedingungen und Haftungsausschlüsse gelten gleichermaßen für die Softwarebibliothek, sowie die in diesem Handbuch angebotenen Beschreibungen und Erläuterungen, auch wenn dies nicht explizit erwähnt wird.

1.2. Lizenzbedingungen

Die Verwendung der OSCAT Bibliothek ist kostenfrei und kann ohne Lizenzvereinbarung für private oder gewerbliche Zwecke eingesetzt werden. Eine Verbreitung der Bibliothek ist ausdrücklich erwünscht, hat aber kostenfrei und unter Hinweis auf unsere Webpage WWW.OSCAT.DE zu erfolgen. Wird die Bibliothek in

elektronischer Form zum Download bereitgestellt oder auf Datenträgern verbreitet, so ist sicherzustellen, dass ein deutlich erkennbarer Hinweis auf OSCAT und ein Weblink zu WWW.OSCAT.DE entsprechend enthalten sind.

1.3. Bestimmungsgemäßer Gebrauch

Die in der OSCAT Bibliothek enthaltenen und in dieser Dokumentation beschriebenen Softwaremodule sind ausschließlich für Fachpersonal mit einer Ausbildung in SPS Programmierung entwickelt worden. Die Anwender sind verantwortlich für die Einhaltung aller geltenden Normen und Vorschriften, die bei der Anwendung zum Tragen kommen. OSCAT weist weder im Handbuch noch in der Software auf diese Normen und Vorschriften hin.

1.4. Eingetragene Markenzeichen

Alle in dieser Beschreibung benutzten Markennamen werden ohne Verweis auf deren Eintragung beziehungsweise Besitzer verwendet. Die Existenz solcher Rechte kann daher nicht ausgeschlossen werden. Die benutzten Markennamen sind Eigentum des jeweiligen Besitzers. Eine Verwendung der Beschreibung für kommerzielle Zwecke ist daher nicht gestattet, auch nicht auszugsweise.

1.5. Sonstiges

Alle rechtsverbindlichen Regelungen befinden sich ausschließlich im Kapitel 1 des Benutzerhandbuchs. Eine Ableitung oder Bezug von rechtlichen Ansprüchen aufgrund des Inhalts des Manuals, außer den Bestimmungen in Kapitel 1, ist gänzlich ausgeschlossen.

2. Einleitung

2.1. Ziele

OSCAT steht für "Open Source Community for Automation Technology".

OSCAT erstellt eine Open Source Bibliothek nach dem IEC61131-3 Standard, welche auf herstellerspezifische Funktionen verzichtet und deshalb auf alle IEC61131-3 kompatiblen Speicherprogrammierbaren Steuerungen portiert werden kann. Auch wenn Entwicklungen für SPS unter dem Einsatz von herstellerspezifischen Bibliotheken meist effizient zu lösen sind und diese Bibliotheken auch teilweise kostenfrei zur Verfügung gestellt werden, ergeben sich doch große Nachteile durch ihren Einsatz:

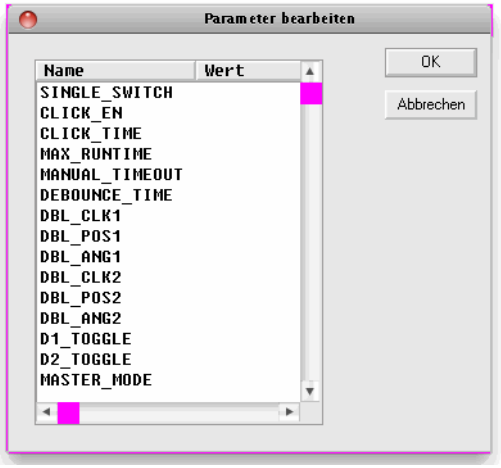
1. Die Bibliotheken der Hersteller sind praktisch alle geschützt und der Source Code ist nicht frei zugänglich, was im Fehlerfall eine Fehlersuche und auch die Behebung des Fehlers enorm schwierig, oft sogar unmöglich macht.
2. Die grafische Erstellung von Programmen kann mit herstellerspezifischen Bibliotheken schnell unübersichtlich, ineffizient und fehleranfällig werden, weil vorhandene Funktionen wegen des fehlenden Source Codes den eigentlichen Bedürfnissen nicht angepasst und erweitert werden können.
3. Ein Wechsel der Hardware, insbesondere der Wechsel zu einem anderen Hersteller, ist durch die geschützten Bibliotheken verhindert und die Vorteile, die ein Standard wie IEC61131 bieten würde, werden so eingeschränkt. An einen Austausch einer herstellerspezifischen Bibliothek mit der eines Wettbewerbers ist ausgeschlossen, denn die Bibliotheken der Hersteller unterscheiden sich enorm in Umfang und Inhalt.
4. Das Verständnis komplexer Module ist ohne einen Einblick in den Sourcecode oft sehr schwierig. Dadurch werden Programme ineffizient und fehleranfällig.

OSCAT will mit der quell offenen OSCAT Bibliothek einen mächtigen und umfassenden Standard für die Programmierung von SPS schaffen, der im Source Code zur Verfügung steht und durch vielfältige Anwendungen ausführlich verifiziert und getestet wurde. Weiterhin fließen durch die Vielzahl der Anwendungen umfangreiche Erkenntnisse und Anregungen in die Bibliothek ein. Dadurch kann die Bibliothek als sehr praxisnah bezeichnet werden. OSCAT versteht seine Bibliothek als Entwicklungsvorlage und nicht als ausgereiftes Produkt. Der Nutzer ist selbst verantwortlich dafür, eventuell in seiner Anwendung verwendete Module mit geeigneten Verfahren zu testen und die notwendige Fehlerfreiheit, Qualität und Funktionalität zu verifizieren. An dieser Stelle sei noch einmal auf die Lizenz-

bedingungen und den Haftungsausschluss in dieser Dokumentation hingewiesen.

2.2. Konventionen

1. Direkte Manipulationen im Speicher:
   Funktionen, die Eingangswerte über Pointer verändern, wie zum Beispiel _Array_Sort, beginnen alle mit einem Unterstrich „_“. _Array_Sort sortiert ein Array direkt im Speicher, was den gravierenden Vorteil hat, dass ein eventuell sehr großes Array erst gar nicht der Funktion übergeben werden muss und deshalb Speicher in der Größe des Arrays und die Zeit zum Kopieren eingespart wird. Es wird allerdings nur erfahrenen Anwendern empfohlen diese Funktionen einzusetzen, da eine Fehlanwendung zu gravierenden Fehlern und Abstürzen führen kann! Bei Anwendung von Funktionen die mit einem „_“ beginnen ist besondere Sorgfalt angebracht und insbesondere darauf zu achten, dass die Aufrufparameter niemals undefinierte Werte annehmen können.
2. Namensgebung bei Funktionen:
   Funktionsbausteine mit Zeitverhalten, wie etwa die Funktion PT1 werden durch die Namensgebung FT_Bausteinname (FT_PT1) beschrieben. Funktionen ohne Zeitbezug sind mit F_Bausteinname angegeben.
3. Logische Gleichungen:
   Innerhalb dieses Handbuchs werden die logischen Verknüpfungen & für UND bzw. AND, + für ODER bzw. OR, /A für ein negiertes A und # für XOR (exklusives ODER) verwendet.
4. Setup-Werte für Bausteine:
   Damit die Anwendung und Programmierung übersichtlich bleibt und komplexe Funktionen einfacher dargestellt werden können, haben viele der Bausteine der OSCAT Bibliothek einstellbare Parameter, die bei Anwendung durch einen Doppelklick auf das grafische Symbol des Bausteins bearbeitet werden können. Ein Doppelklick auf das Symbol öffnet eine Dialogbox, die das Bearbeiten der Setup-Werte erlaubt. Wird eine Funktion mehrfach verwendet, so können damit je Baustein die Setup-Werte einzeln festgelegt werden. Die Bearbeitung durch Doppelklick funktioniert unter CoDeSys ausschließlich in CFC. In ST müssen alle Parameter,



auch die Setup-Parameter, im Aufruf übergeben werden. Die Setup-Parameter werden einfach nach den normalen Eingängen angefügt. Die Parameter werden in der grafischen Oberfläche durch Doppelklick bearbeitet und dann wie Konstanten unter IEC61131 eingegeben. Es ist Dabei zu beachten dass Zeiten mit T#200ms und TRUE und FALSE in Großbuchstaben geschrieben sein müssen.

5. Error- und Status-Reporting (ESR):

   Komplexere Bausteine erhalten zum großen Teil einen Error- oder Status-Ausgang. Ein Error-Ausgang ist 0, falls kein Fehler bei der Ausführung auftritt. Tritt jedoch im Baustein ein Fehler auf, so nimmt dieser Ausgang einen Wert im Bereich 1 .. 99 an und meldet somit einen Fehler mit Fehlernummer. Ein Status- oder Error-Sammelmodul kann diese Meldungen sammeln und mit einem Zeitstempel versehen, in einer Datenbank bzw. Array speichern, oder per TCP/IP an übergeordnete Systeme weiterleiten. Ein Ausgang des Typs Status ist kompatibel zu einem Error-Ausgang mit identischer Funktion. Jedoch meldet ein Status-Ausgang nicht nur Fehler, sondern führt auch über Aktivitäten des Bausteins Protokoll. Werte zwischen 1 .. 99 sind weiterhin Fehlermeldungen. Zwischen 100 .. 199 befinden sind Meldungen über Zustandsveränderungen. Der Bereich 200 .. 255 ist reserviert für Debug-Meldungen. Mit dieser, innerhalb der OSCAT Bibliothek standardisierten Funktionalität, wird eine einfache und übergreifende Möglichkeit geboten, Betriebszustandsmeldungen und Fehlermeldungen auf einfache Weise zu integrieren, ohne die Funktion eines Systems zu beeinflussen. Bausteine, die dieses Verfahren unterstützen, werden ab der Revision 1.4 mit der Kennzeichnung „ESR-Fähig“ gekennzeichnet. Weitere Informationen zu den ESR-Bausteinen finden Sie im Abschnitt „Diverse Funktionen“.

2.3. Testumgebung und Voraussetzungen

Verfügbare Plattformen und damit verbundene Abhängigkeiten

CoDeSys:

Benötigt die Bibliothek „SysLibFile.lib“ und „SysLibSockets.lib“

Lauffähig auf

WAGO 750-841

CoDeSys SP PLCWinNT V2.4

und kompatible Plattformen

PCWORX:

Keine zusätzliche Bibliothek notwendig

Lauffähig auf allen Steuerungen mit Filesystem und Ethernet ab Firmware >= 3.5x

BECKHOFF:

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0 oder höher	PC or CX (x86)	TcSystem.Lib TcBase.Lib TcSystem.Lib
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	TcSystem.Lib TcBase.Lib TcSystem.Lib

Erfordert die Installation des„TwinCAT TCP/IP Connection Server“
Benötigt somit die Bibliothek „TcpIp.Lib“
(Standard.Lib; TcBase.Lib; TcSystem.Lib werden danach automatisch eingebunden)

Programmierumgebung:
NT4, W2K, XP, Xpe;
TwinCAT System Version 2.8 oder höher;
TwinCAT Installation Level: TwinCAT PLC oder höher;
Zielplattform:
TwinCAT SPS-Laufzeitsystem Version 2.8 oder höher.
PC or CX (x86)

TwinCAT TCP/IP Connection Server v1.0.0.0 oder höher;
NT4, W2K, XP, XPe, CE (image v1.75 oder höher);
CX (ARM)

TwinCAT TCP/IP Connection Server v1.0.0.44 oder höher;
CE (image v2.13 oder höher);

2.4. Aktualität

OSCAT aktualisiert diese Beschreibung fortlaufend. Es wird empfohlen sich die jeweils aktuellste Version von der OSCAT Homepage unter www.OSCAT.DE zu laden. Hier wird das jeweils aktuellste Manual zum Download bereitgestellt. Neben dem Manual stellt OSCAT auch eine detaillierte Revision Historie bereit. Die "OSCAT Revision History" listet alle Revisionen der einzelnen Bausteine mit Änderungen und ab welcher Release der Bibliothek dieser Baustein enthalten ist.

2.5. Support

Support wird durch die Vielzahl der Anwender selbst im Forum unter WWW.OSCAT.DE zur Verfügung gestellt. Ein Anspruch auf Support besteht aber generell nicht, auch dann nicht, wenn sich herausstellen sollte, dass die Bibliothek oder Teile der Bibliothek fehlerhaft sind. Der im Rahmen des OSCAT Forums bereitgestellte Support wird von Anwendern freiwillig und untereinander bereitgestellt. Updates der Bibliothek und der Dokumentation werden in der Regel einmal im Monat auf der Homepage von OSCAT unter WWW.OSCAT.DE zur Verfügung gestellt. Ein Anspruch auf Wartung, Fehlerbehebung und Softwarepflege jedweder Art besteht generell nicht und ist als freiwillige Leistung von OSCAT anzusehen. Bitte senden Sie bei Support Anfragen keine email an OSCAT. Anfragen können schneller und effektiver bearbeitet werden wenn die Anfragen in unserem Forum gestellt werden.

3. Demo-Programme

3.1. Demo-Programme

Die OSCAT Network Bibliothek beinhaltet Bausteine und Funktionalitäten die sich mit dem Thema Datei-Handling und Ethernet-Kommunikation befassen, und mitunter besonderes Basis-Wissen voraussetzen. Um hier den Anwender einen möglichst einfachen Einstieg zu ermöglichen, sind für viele Themen Demo-Programme vorbereitet.

Die Demo-Programme sind in der network.lib im Ordner „DEMO“ enthalten. Wenn diese benutzt werden, sollten die benötigten Programme in das eigene Projekt kopiert und den eigenen Bedürfnissen angepasst werden. Manche Bausteine erfordern die Angabe von eigenen spezifischen Parametern, damit diese voll funktionieren.

In der Codesys und Beckhoff Bibliothek sind die Demo-Programme ausgeblendet, da sie ansonsten unnötig Ressourcen belegen würden.

BASE64_DEMO

CSV_PARSER_BUF_DEMO

CSV_PARSER_FILE_DEMO

DLOG_FILE_CSV

DLOG_FILE_CSV_FTP_DEMO

DLOG_FILE_CSV_SMTP_DEMO

DLOG_FILE_HTML_DEMO

DLOG_FILE_XML_DEMO

DLOG_MYSQL_DEMO

DLOG_RRD_DEMO

DNS_DYN_DEMO

DNS_REV_DEMO

DNS_SNTP_SYSLOG_DEMO

FILE_BLOCK_DEMO

FTP_CLIENT_DEMO

GET_WAN_IP_DEMO

HTTP_DEMO

INI_PARSER_BUF_DEMO

INI_PARSER_FILE_DEMO
IP2GEO_DEMO
IRTRANS_DEMO
MB_CLIENT_DEMO
MB_SERVER_DEMO
MD5_CRAM_AUTH_DEMO
NET_VAR_MASTER_DEMO
NET_VAR_SLAVE_DEMO
RC4_CRYPT_DEMO
SHA_MD5_DEMO
SMTP_CLIENT_DEMO
SPIDER_DEMO
TELNET_LOG_DEMO
TELNET_PRINT_DEMO
TN_VISION_DEMO_1
TN_VISION_DEMO_2
YAHOO_WEATHER_DEMO
WORLD_WEATHER_DEMO

4. Datentypen der NETWORK-Bibliothek

4.1. DLOG_DATA

Die Struktur DLOG_DATA dient zur Kommunikation der DLOG_* Bausteine.

DLOG_DATA:

Datenfeld	Datentyp	Beschreibung
STORE_TYPE	BYTE	Typ des DLOG_STORE Bausteins
ADD_COM	INT	Daten speichern Kommando
ADD_DATA_REQ	BOOL	Daten speichern von Extern
CLOCK_TRIG	BOOL	DTI (Date-Time) neuer Wert
ID_MAX	USINT	Anzahl der DLOG_* Bausteine
DTI	DT	aktueller Date-Time Wert
UCB	UNI_CIRCULAR_BUFFER_DATA	Datenspeicher
NEW_FILE_STRING	STRING	Name der neuen Datei
NEW_FILE_SIZE	UDINT	Größe der neuen Datei
NEW_FILE_RTRIG	BOOL	Flanke neue Datei wurde erstellt

4.2. us_LOG_VIEWPORT

us_LOG_VIEWPORT:

Name	Typ	Eigenschaften
LINE_ARRAY	ARRAY [1..40] OF INT	LOG-Index Verweise
COUNT	INT	Anzahl der sichtbaren Nachtrichten
UPDATE_COUNT	UINT	Update-Zähler
MOVE_TO_X	INT	Steuerung der Nachrichtenanzeige
UPDATE	BOOL	Daten wurden geändert -> neu zeichnen

4.3. URL

Die Struktur URL speichert die einzelnen Bestandteile einer URL.

URL:

Datenfeld	Datentyp	Beschreibung
PROTOCOL	STRING(10)	Protokoll
USER	STRING(32)	User Name
PASSWORD	STRING(32)	Passwort
DOMAIN	STRING(80)	Domain
PORT	WORD	Port Nummer
PATH	STRING(80)	Pfadangabe
QUERY	STRING(120)	Query
ANCHOR	STRING(40)	Anker
HEADER	STRING(160)	Header

4.4. us_TN_INPUT_CONTROL

Eine Variable vom Typ us_TN_INPUT_CONTROL kann benutzt werden um verschiedene INPUT_CONTROL Elemente zu parametrieren und zu verwalten, sowie zur Darstellung der ToolTip Info.

us_TN_INPUT_CONTROL:

Datenfeld	Datentyp	Beschreibung
bo_Enable	BOOL	Bearbeitung freigeben / sperren
bo_Update_all	BOOL	Alle Elemente neu zeichnen
bo_Reset_Fokus	BOOL	Fokus auf 1. Element setzen
in_Fokus_at	INT	Element mit aktivem Fokus
in_Count	INT	Anzahl der INPUT_CONTROL Elemente
in_ToolTip_X	INT	ToolTip Text X-Offset
in_ToolTip_Y	INT	ToolTip Text Y-Offset
by_ToolTip_Attr	BYTE	ToolTip Text Attribute (Farbe)

in_ToolTip_Size	INT	ToolTip Text Länge
usa_TN_INPUT_CONTROL_DATA	ARRAY [1..20] OF us_TN_INPUT_CONTROL_DATA	

4.5. us_TN_INPUT_CONTROL_DATA

Eine Variable vom Typ us_TN_INPUT_CONTROL_DATA kann benutzt werden um ein INPUT_CONTROL Element zu parametrieren, und Element bezogenen Eingaben / Ereignisse zu verarbeiten.

us_TN_INPUT_CONTROL_DATA:

Datenfeld	**Datentyp**	**Beschreibung**
by_Input_Exten_Code	BYTE	Tastencode Extended
by_Input_ASCII_Code	BYTE	Tastencode ASCII
bo_Input_ASCII_IsNum	BOOL	Tastencode ist eine Ziffer
in_Title_X_Offset	INT	Title Text X-Offset
in_Title_Y_Offset	INT	Title Text Y-Offset
by_Title_Attr	BYTE	Title Text Attribute
st_Title_String	STRING	Title Text
in_Cursor_X	INT	aktuelle Cursor X-Position
in_Cursor_Y	INT	aktuelle Cursor Y-Position
in_Type	INT	Element Type
in_X	INT	Element X-Position
in_Y	INT	Element Y-Position
in_Cursor_Pos	INT	aktuelle Cursor-Position
by_Attr_mF	BYTE	Attribute für Element mit Fokus
by_Attr_oF	BYTE	Attribute für Element ohne Fokus
in_selected	INT	Text Element wurde angewählt
st_Input_Mask	STRING	Eingabemaske

st_Input_Data	STRING(STRING_LENGTH)	Text aktuelle Eingabe
st_Input_String	STRING	Text Kopie nach Eingabe
st_Input_ToolTip	STRING	Text für ToolTip
in_Input_Option	INT	Text Optionen
bo_Input_Entered	BOOL	Text mit RETURN-Taste übergeben
bo_Input_Hidden	BOOL	Text versteckte Eingabe mit '*'
bo_Input_Only_Num	BOOL	Text nur Nummerneingabe zulassen
bo_Focus	BOOL	Element besitzt den Fokus
bo_Update_Input	BOOL	Element wegen Eingabe neu zeichnen
bo_Update_All	BOOL	Element komplett neu zeichnen

4.6. us_TN_MENU

Eine Variable vom Typ us_TN_MENU kann benutzt werden um ein MENU Element zu parametrieren, zur Anzeige zu bringen und Element bezogenen Eingaben zu verarbeiten.

us_TN_MENU:

Datenfeld	**Datentyp**	**Beschreibung**
st_Menu_Text	STRING(STRING_LENGTH)	Menü Elemente
in_Menu_E_Count	INT	Anzahl der Menü-Elemente
in_Y	INT	Menü Y-Position
in_X	INT	Menü X-Position
by_Attr_mF	BYTE	Text Attribute mit Fokus
by_Attr_oF	BYTE	Text Attribute ohne Fokus
in_X_SM_new	INT	Sub-Menü neue X-Position
in_Y_SM_new	INT	Sub-Menü neue Y-Position
in_X_SM_old	INT	Sub-Menü alte X-Position
in_Y_SM_old	INT	Sub-Menü alte Y-Position
in_Cur_Menu_Item	INT	aktuelles Haupt-Menü-Element

in_Cur_Sub_Item	INT	aktuelles Sub-Menü-Element
in_State	INT	Menü Status
in_Menu_Selected	INT	ausgewähltes Menü Element
bo_Create	BOOL	Aktion: Menü neu erzeugen
bo_Destroy	BOOL	Aktion: Menü entfernen
bo_Update	BOOL	Aktion: Menü aktualisieren

4.7. us_TN_MENU_POPUP

Eine Variable vom Typ us_TN_MENU_POPUP kann benutzt werden um ein POPUP-MENU Element zu parametrieren, zur Anzeige zu bringen und Element bezogenen Eingaben zu verarbeiten.

us_TN_MENU_POPUP:

Datenfeld	**Datentyp**	**Beschreibung**
st_Menu_Text	STRING(STRING_LENGTH)	Menü Elemente
in_Menu_E_Count	INT	Anzahl der Menü-Elemente
in_X	INT	Menü X-Position
in_Y	INT	Menü Y-Position
in_Cols	INT	Menü Anzahl der Spalten
in_Rows	INT	Menü Anzahl der Zeilen
in_Cur_Item	INT	aktuelles Menü-Element
by_Attr_mF	BYTE	Text Attribute mit Fokus
by_Attr_oF	BYTE	Text Attribute ohne Fokus
by_Input_Exten_Code	BYTE	Tastencode - Sondertasten
bo_Create	BOOL	Aktion: Menü neu erzeugen
bo_Destroy	BOOL	Aktion: Menü entfernen
bo_Update	BOOL	Aktion: Menü aktualisieren
bo_Activ	BOOL	Menü ist aktiv

4.8. us_TN_SCREEN

Eine Variable vom Typ us_TN_SCREEN kann benutzt werden um die grafische Oberfläche (GUI) zu verwalten und zur Anzeige zu bringen

us_TN_SCREEN:

Datenfeld	Datentyp	Beschreibung
bya_CHAR	ARRAY [0..1919] OF BYTE	Screen Charakters
bya_COLOR	ARRAY [0..1919] OF BYTE	Screen Farbcodes
bya_BACKUP	ARRAY [0..1919] OF BYTE	Screen Backup Speicher
bya_Line_Update	ARRAY [0..23] OF BYTE	Screen Zeilen-Updates
by_Input_Exten_Code	BYTE	Tastencode Sondertasten
by_Input_ASCII_Code	BYTE	Tastencode ASCII
bo_Input_ASCII_IsNum	BOOL	Tastencode ist eine Ziffer
in_Page_Number	INT	aktuelle Seiten-Nummer
in_Cursor_X	INT	Cursor X-Position
in_Cursor_Y	INT	Cursor Y-Position
in_EOS_Offset	INT	End of String Offset
by_Clear_Screen_Attr	BYTE	Bildschirm Lösch-Farbe
bo_Clear_Screen_Attr	BOOL	Bildschirm löschen
bo_Modul_Dialog	BOOL	Modaler Dialog aktiv
bo_Menu_Bar_Dialog	BOOL	Menü Dialog aktiv

4.9. FILE_PATH_DATA

Die Struktur FILE_PATH_DATA dient dem Baustein FILE_PATH_SPLIT zum ablegen der einzelnen Elemente.

FILE_PATH_DATA:

Datenfeld	**Datentyp**	**Beschreibung**
DRIVE	STRING(3)	Laufwerksname
DIRECTORY	STRING(STRING_LENGTH)	Verzeichnisname
FILENAME	STRING	Dateiname

4.10. FILE_SERVER_DATA

FILE_SERVER Datenstruktur:

Name	Typ	Eigenschaften
FILE_OPEN	BOOL	Datei ist geöffnet
FILENAME	STRING	Dateiname
MODE	BYTE	Betriebsart – Kommando
OFFSET	UDINT	Datei-Offset für Lesen und Schreiben
FILE_SIZE	UDINT	Aktuelle Größe der Datei in Bytes
ERROR	BYTE	Fehlercodes (System abhängig)

4.11. IP2GEO

IP2GEO Datenstruktur:

Name	Typ	Eigenschaften
STATE	BOOL	Daten sind gültig
IP4	DWORD	IP-Adresse der geographischen Daten
COUNTRY_CODE	STRING(2)	Code des Landes (ISO Format) z.B. AT = Austria

COUNTY_NAME	STRING(20)	Name des Landes
REGION_CODE	STRING(2)	Code der Region (FIPS Format) z.B. 09 = Vienna
REGION_NAME	STRING(20)	Name der Region
CITY	STRING(20)	Name der Stadt
GEO_LATITUDE	REAL	Breitengrad des Ortes
GEO_LONGITUDE	REAL	Längengrad des Ortes
TIME_ZONE_NAME	STRING(20)	Name der Zeitzone
GMT_OFFSET	INT	Offset zu Universelle Weltzeit in Sekunden
IS_DST	BOOL	Sommerzeit aktiv

4.12. IP_C

IP_C Datenstruktur:

Datenfeld	**Datentyp**	**Beschreibung**
C_MODE	BYTE	Type der Verbindung
C_PORT	WORD	Port-Nummer
C_IP	DWORD	kodierte IP v4 Adresse
C_STATE	BYTE	Status der Verbindung
C_ENABLE	BOOL	Verbindungsfreigabe
R_OBSERVE	BOOL	Datenempfang überwachen
TIME_RESET	BOOL	Rücksetzen der Überwachungszeiten
ERROR	DWORD	Fehlercode
FIFO	IP_FIFO_DATA	Datenstruktur der Zugriffsverwaltung (kein Anwender-Zugriff notwendig)
MAILBOX	ARRAY [1..16] OF BYTE	Mailbox: Datenbereich für Bausteindatenaustausch

4.13. IP_FIFO_DATA

IP_FIFO_DATA Datenstruktur:

Datenfeld	Datentyp	Beschreibung
X	ARRAY [1..128] OF BYTE	FIFO-Speicher mit eingetragenen ID's
Y	ARRAY [1..128] OF BYTE	Anzahl der Einträge pro ID's
ID	BYTE	Zuletzt vergebene ID (Höchste ID)
MAX_ID	BYTE	Maximale Anzahl von Anmeldungen pro ID
INIT	BOOL	Initialisierung durchgeführt
EMPTY	BOOL	FIFO ist leer
FULL	BOOL	FIFO ist voll (sollte nicht passieren !)
TOP	INT	Maximale Anzahl an Einträgen in FIFO
NW	INT	Schreib-Index FIFO
NR	INT	Lese-Index FIFO

4.14. LOG_CONTROL

us_LOG_VIEWPORT Datenstruktur:

Name	Typ	Eigenschaften
NEW_MSG	STRING(STRING_LENGTH)	Neue Meldung - Text
NEW_MSG_OPTION	DWORD	Neue Meldung – Option BYTE 3: Reserve BYTE 2: Level BYTE 1: Backcolor BYTE 0: Frontcolor
LEVEL	BYTE	Vorgegebener Log-Level
SIZE	INT	Größe des Array (maximaler Index)
RESET	BOOL	Rücksetzen / Löschen der Einträge
PRINTF	ARRAY[1.11] OF STRING(STRING_LENGTH)	Parameterdaten für PRINT_SF Baustein
MSG	ARRAY[0.N] OF	Array für Meldungen - Text

	STRING(STRING_LENGTH)	
MSG_OPTION	ARRAY[0.N] OF DWORD	Array für Meldungen – Option BYTE 3: Reserve BYTE 2: Level BYTE 1: Backcolor BYTE 0: Frontcolor
UPDATE_COUNT	UINT	Update-Zähler (wird mit jeder neuen Nachricht erhöht)
IDX	INT	Aktueller Ausgabe-Index
RING_MODE	BOOL	BUFFER Überlauf / Ringmode aktiviert

4.15. NET_VAR_DATA

NET_VAR Datenstruktur:

Name	Typ	Eigenschaften
CYCLE	UDINT	Zyklus Zähler
STATE	BYTE	Betriebszustand
INDEX	INT	Lese/Schreibindex
ID_MAX	USINT	Anzahl der Satellitenbausteine
ERROR_ID	BYTE	ID-Nummer des gestörten Bausteines
BUF_SIZE	UINT	Größe des Datenpuffer (Bytes)
S_BUF	NETWORK_BUFFER	Netzwerk-Puffer für Daten Senden
R_BUF	NETWORK_BUFFER	Netzwerk-Puffer für Daten Empfangen

4.16. PRINTF_DATA

PRINTF_DATA Datenstruktur:

Datenfeld	**Datentyp**	Beschreibung

PRINTF	ARRAY [1..11] OF STRING(LOG_SIZE)	Array für Parameterübergabe

4.17. UNI_CIRCULAR_BUFFER_DATA

Die Struktur UNI_CIRCULAR_BUFFER_DATA dient zur Datenverwaltung für den Baustein UNI_CIRCULAR_BUFFER

UNI_CIRCULAR_BUFFER_DATA:

Datenfeld	**Datentyp**	**Beschreibung**
D_MODE	INT	MODE (Kommandonummer)
D_HEAD	WORD	Header-Information Lesen/Schreiben
D_STRING	STRING(STRING_LENGTH)	STRING Lesen/Schreiben
D_REAL	REAL	REAL Lesen/Schreiben
D_DWORD	DWORD	DWORD Lesen/Schreiben
BUF_SIZE	UINT	Anzahl Bytes im Buffer
BUF_COUNT	UINT	Anzahl der Elemente im Buffer
BUF_USED	USINT	Füllstand (0-100 %)
BUF	NW_BUF_LONG	Daten-BUFFER
_GetStart	UINT	Intern: Lesezeiger
_GetEnd	UINT	Intern: Lesezeiger
_Last	UINT	Intern: Datenzeiger
_First	UINT	Intern: Datenzeiger

4.18. VMAP_DATA

VMAP_DATA Datenstruktur:

Name	Typ	Eigenschaften
FC	DWORD	Funktionscode: Freigabe-Bitmaske

V_ADR	INT	Virtueller Adressbereich: Startadresse
V_SIZE	INT	Virtueller Adressbereich: Anzahl Word
P_ADR	INT	Realer Adressbereich: Startadresse
TIME_OUT	TIME	Watchdog

4.19. XML_CONTROL

XML_CONTROL Datenstruktur:

Datenfeld	Datentyp	Beschreibung
COMMAND	WORD	Steuerbefehle (Binäre Belegung)
START_POS	UINT	(Puffer-Index des ersten Zeichen)
STOP_POS	UINT	(Puffer-Index des letzten Zeichen)
COUNT	UINT	Elementnummer
TYP	INT	Type-Code des aktuellen Element
LEVEL	UINT	Aktuelle Hierarchie / Ebene
PATH	STRING(STRING_LENGTH)	Hierarchie als TEXT (PFAD)
ELEMENT	STRING(STRING_LENGTH)	Aktuelles Element als TEXT
ATTRIBUTE	STRING(STRING_LENGTH)	Aktuelles Attribute als TEXT
VALUE	STRING(STRING_LENGTH)	Aktueller Wert als TEXT
BLOCK1_START	UINT	Start-Position von Block 1
BLOCK1_STOP	UINT	Stopp-Position von Block 1
BLOCK2_START	UINT	Start-Position von Block 2
BLOCK2_STOP	UINT	Stopp-Position von Block 2

4.20. WORLD_WEATHER_DATA

WORLD_WEATHER_DATA Datenstruktur:

Name	Typ	Eigenschaften
CUR	WORLD_WEATHER_CUR	Current weather conditions
DAY	ARRAY [0..4] OF WORLD_WEATHER_DAY	Next 5 days of weather forecast

WORLD_WEATHER_CUR Datenstruktur:

Name	Typ	Eigenschaften
OBSERVATION_TIME	STRING(8)	Observation time (UTC)
TEMP_C	INT	Temperature (°C)
WEATHER_CODE	INT	Unique Weather Code
WEATHER_DESC	STRING(60)	Weather description text
WEATHER_ICON	INT	Weather Icon
WIND_SPEED_MILES	INT	Wind speed in miles per hour
WIND_SPEED_KMPH	INT	Wind speed in kilometre per hour
WIND_DIR_DEGREE	INT	Wind direction in degree
WIND_DIR16POINT	STRING(3)	16-Point wind direction compass
PRECIPMM	REAL	Precipitation amount in millimetre
HUMIDITY	INT	Humidity (%)
VISIBILITY	INT	Visibility (km)
PRESSURE	INT	Atmospheric pressure in milibars
CLOUDOVER	INT	Cloud cover (%)

WORLD_WEATHER_DAY Datenstruktur:

Name	Typ	Eigenschaften
DATE_OF_DAY	STRING(10)	Date for which the weather is forecasted

TEMP_MAX_C	INT	Day temperature in °C(Celcius)
TEMP_MAX_F	INT	Day temperature in °F(Fahrenheit)
TEMP_MIN_C	INT	Night temperature in °C(Celcius)
TEMP_MIN_F	INT	Night temperature in °F(Fahrenheit)
WIND_SPEED_MILES	INT	Wind Speed in mph (miles per hour)
WIND_SPEED_KMPH	INT	Wind Speed in kmph (Kilometer per hour)
WIND_DIR_DEGREE	INT	Wind direction in degree
WIND_DIR16POINT	STRING(3)	16-Point wind direction compass
WEATHER_CODE	INT	A unique weather condition code
WEATHER_DESC	STRING(60)	Weather description text
WEATHER_ICON	INT	Weather Icon
PRECIPMM	REAL	Precipitation Amount (millimetre)

4.21. YAHOO_WEATHER_DATA

YAHOO_WEATHER Datenstruktur:

Name	Typ	Eigenschaften
TimeToLive	INT	Time to Live: how long in minutes this feed should be cached
location_city	STRING(40)	The location of this forecast: city: city name
location_region	STRING(20)	The location of this forecast: region: state, territory, or region, if given
location_country	STRING(20)	The location of this forecast: country:
unit_temperature	STRING(1)	temperature: degree units, f for Fahrenheit or c for Celsius
unit_distance	STRING(2)	distance: units for distance, mi for miles or km for kilo-

		meters
unit_pressure	STRING(2)	pressure: units of barometric pressure, in for pounds per square inch or mb for millibars
unit_speed	STRING(3)	speed: units of speed, mph for miles per hour or kph for kilometers per hour
wind_chill	INT	Forecast information about wind chill in degrees
wind_direction	INT	Forecast information about wind direction, in degrees
wind_speed	REAL	Forecast information about wind speed, in the units (mph or kph)
atmosphere_humidity	INT	Forecast information about current atmospheric humidity: humidity, in percent
atmosphere_pressure	INT	Forecast information about current atmospheric pressure: barometric pressure, in the units (in or mb)
atmosphere_visibility	REAL	Forecast information about current atmospheric visibility, in the units (mi or km)
atmosphere_rising	INT	Forecast information about rising: state of the barometric pressure: steady (0), rising (1), or falling (2). (integer: 0, 1, 2)
astronomy_sunrise	STRING(10)	sunrise: today's sunrise time. The time is a string in a local time format of "h:mm am/pm"
astronomy_sunset	STRING(10)	sunset: today's sunset time. The time is a string in a local time format of "h:mm am/pm"
geo_latitude	REAL	The latitude of the location
geo_longitude	REAL	The longitude of the location
cur_conditions_temp	INT	The current weather conditions: temp: the current temperature, in the units (f or c)
cur_conditions_text	STRING(40)	The current weather conditions: text: a textual description of conditions
cur_conditions_code	INT	The current weather conditions: code: the condition code for this forecast
cur_conditions_icon	INT	The current weather conditions: icon: the condition icon for this forecast
forcast_today_low_temp	INT	The forcast today weather conditions: the forecasted low temperature for this day in the units (f or c)
forcast_today_high_temp	INT	The forcast today weather conditions: the forecasted high temperature for this day in the units (f or c)
forcast_today_text	STRING(40)	The forcast today weather conditions: text: a textual de-

		scription of conditions
forcast_today_code	INT	The current weather conditions: code: the condition code for this forecast
forcast_today_icon	INT	The current weather conditions: icon: the condition icon for this forecast
forcast_tomorrow_low_temp	INT	The forcast tomorrow weather conditions: the forecasted low temperature for this day in the units (f or c)
forcast_tomorrow_high_temp	INT	The forcast tomorrow weather conditions: the forecasted high temperature for this day in the units (f or c)
forcast_tomorrow_text	STRING(40)	The forcast tomorrow weather conditions: text: a textual description of conditions
forcast_tomorrow_code	INT	The current weather conditions: code: the condition code for this forecast
forcast_tomorrow_icon	INT	The current weather conditions: icon: the condition icon for this forecast

Condition Codes:

Die Felder cur_conditions_code, forcast_today_code und forcast_tomorrow_code beschreiben die Wetterlage in Textform durch Angabe eines „Condition Codes“

Wert	Beschreibung
0	tornado
1	tropical storm
2	hurricane
3	severe thunderstorms
4	thunderstorms
5	mixed rain and snow
6	mixed rain and sleet
7	mixed snow and sleet
8	freezing drizzle
9	drizzle
10	freezing rain

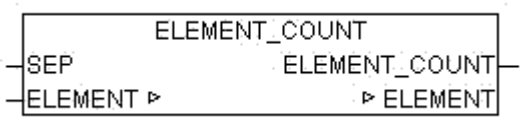
11	showers
12	showers
13	snow flurries
14	light snow showers
15	blowing snow
16	snow
17	hail
18	sleet
19	dust
20	foggy
21	haze
22	smoky
23	blustery
24	windy
25	cold
26	cloudy
27	mostly cloudy (night)
28	mostly cloudy (day)
29	partly cloudy (night)
30	partly cloudy (day)
31	clear (night)
32	sunny
33	fair (night)
34	fair (day)
35	mixed rain and hail
36	hot
37	isolated thunderstorms
38	scattered thunderstorms

39	scattered thunderstorms
40	scattered showers
41	heavy snow
42	scattered snow showers
43	heavy snow
44	partly cloudy
45	thundershowers
46	snow showers
47	isolated thundershowers
3200	not available

5. Diverse Funktionen

5.1. ELEMENT_COUNT

Type	Funktion : INT
Input	SEP : BYTE (Separationszeichen der Elemente)
I/O	ELEMENT : STRING(ELEMENT_LENGTH) (Eingangsliste)
Output	INT (Anzahl der Elemente in der Liste)



ELEMENT_COUNT ermittelt die Anzahl der Elemente einer Liste.

Ist der Parameter ELEMENT ein leerer String so wird als Ergebnis 0 ausgegeben. Befindet sich mindestens ein Zeichen in ELEMENT wird dies als einzelnes Element bewertet und als ELEMENT_COUNT = 1 ausgegeben.

Beispiele:

ELEMENT_COUNT('0,1,2,3',44) = 4

ELEMENT_COUNT('',44) = 0

ELEMENT_COUNT('x',44) = 1

5.2. ELEMENT_GET

Type	Funktion : STRING(ELEMENT_LENGTH)
Input	SEP : BYTE (Separationszeichen der Elemente)
	POS : INT (Position des Elements)
I/O	ELEMENT : STRING(ELEMENT_LENGTH) (Eingangsliste)
Output	STRING (Ausgangsstring)

ELEMENT_GET lieferte das Element an der Stelle POS aus einer Elementeinliste. Die Liste besteht aus Zeichenketten die mit dem Separationszeichen SEP getrennt sind. Das erste Element der Liste hat die Position 0.

Beispiele:

ELEMENT_GET('ABC,23,,NEXT', 44, 0) = 'ABC'

ELEMENT_GET('ABC,23,,NEXT', 44, 1) = '23'

ELEMENT_GET('ABC,23,,NEXT', 44, 2) = ''

ELEMENT_GET('ABC,23,,NEXT', 44, 3) = 'NEXT'

ELEMENT_GET('ABC,23,,NEXT', 44, 4) = ''

ELEMENT_GET('', 44, 0) = ''

5.3. NETWORK_VERSION

Type Funktion : DWORD

Input IN : BOOL (wenn TRUE liefert der Baustein das Release Datum)

Output (Version der Bibliothek)

NETWORK_VERSION gibt wenn IN = FALSE die aktuelle Versionsnummer als DWORD zurück. Wird IN auf TRUE gesetzt so wird das Release Datum der aktuellen Version als DWORD zurückgegeben.

Beispiel: NETWORK_VERSION(FALSE) = 111 für Version 1.11

DWORD_TO_DATE(NETWORK_VERSION(TRUE)) = 2011-2-3

6. Geräte Treiber

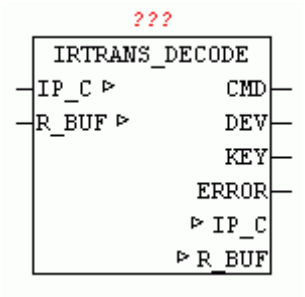
6.1. IRTRANS

Die Bausteine IRTRANS_? stellen ein Interface für Infrarot Transmitter der Firma IRTrans GmbH zur Verfügung. IRTrans bietet Transmitter für RS232 und TCP/IP an, welche alle mit den folgenden Treiberbausteinen betrieben werden können. Der grundsätzliche Anschluss an RS232 oder TCP/IP hat mit den entsprechenden Herstellerroutinen zu erfolgen. Die Interfacebausteine setzen auf ein Buffer Interface auf welches in einen Buffer (Array of Byte) die Daten und in einem Counter die Länge des Datenpakets in Bytes zur Verfügung stellt. Die IRTrans Geräte erlernen beliebige IR Tastencodes und setzten diese mittels einer konfigurierbaren Datenbank in ASCII Strings um. Mit der Ethernet Variante werden diese Strings dann über UDP versandt und können von einer SPS empfangen und ausgewertet werden. So können zum Beispiel vollautomatisch die Jalousien heruntergefahren werden wenn jemand den Fernseher einschaltet ohne das dafür eine zusätzliche Bedienung nötig wäre. Die SPS kann über Diesen Weg beliebig vielen Fernsteuerungen in unterschiedlichen Räumen zuhören und entsprechende Aktionen daraus ableiten. Umgekehrt ist natürlich auch die Aussendung von Tastencodes über die Transmitter Module möglich.

6.2. IRTRANS_DECODE

Type	Funktionsbaustein
I/O	IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)
	R_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT'
	(Empfangsdaten)
Output	CMD : BOOL (TRUE wenn gültige Daten am Ausgang anliegen)
	DEV : STRING (Name der Fernsteuerung)
	KEY : STRING (Name des Tastencodes)
	ERROR : BOOL (TRUE wenn ein ungültiges Datenpaket vorliegt)

IRTRANS_DECODE empfängt die in BUFFER vorliegenden Daten vom Baustein IRTRANS_SERVER, überprüft ob ein gültiges Datenpaket vorliegt und dekodiert aus dem Datenpaket den Namen der Fernsteuerung und den Namen der Taste. Wenn ein gültiges Datenpaket dekodiert wurde liegt der



Name der Fernsteuerung am Ausgang DEV und der Name der Taste am Ausgang KEY. Der Ausgang CMD signalisiert das neue Ausgangsdaten vorliegen. Der Ausgang ERROR wird dann gesetzt wenn ein Datenpaket empfangen wurde das nicht dem Format entspricht.

Das Format ist wie folgt definiert:

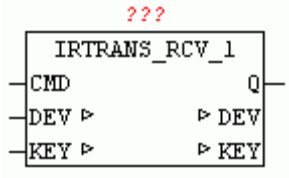
'Name der Fernsteuerung','Name des Tastencodes'RN

Ein Datenpaket besteht aus dem Namen der Fernsteuerung, gefolgt von einem Komma und anschließend der Name des Tastencodes. Das Datenpaket wird mit einem Carriage Return und einem Line Feed abgeschlossen.

Damit IRTRANS_DECODE funktioniert muss in der IRTrans Konfiguration die Check box BROADCAST IR RELAY angekreuzt sein und in der entsprechenden Device Datenbank unter DEFAULT ACTION muss der String '%r,%c\r\n' eingetragen sein. IRTRANS_DECODE wertet genau diesen String aus und dekodiert %r als Name der Fernbedienung und %c als die gedrückte Taste.

6.3. IRTRANS_RCV_1

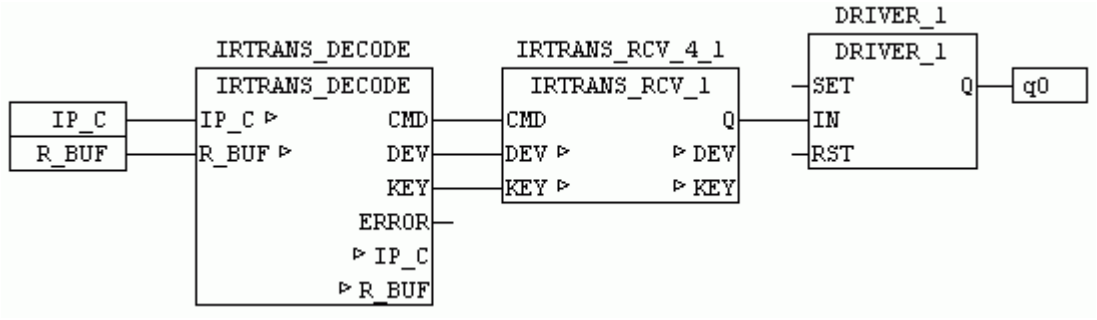
Type	Funktionsbaustein
Input	CMD : BOOL (TRUE wenn Daten zum Auswerten Anliegen)
I/O	DEV : STRING (Name der Fernsteuerung)
	KEY : STRING (NAME der Taste)
Setup	DEV_CODE : STRING (zu dekodierender Fernsteuerungsname)
	KEY_CODE : STRING (zu dekodierender Tastencode)
Output	Q : BOOL (Ausgangssignal)



IRTRANS_RCV_1 überprüft wenn CMD = TRUE ob die Zeichenkette am Ein-

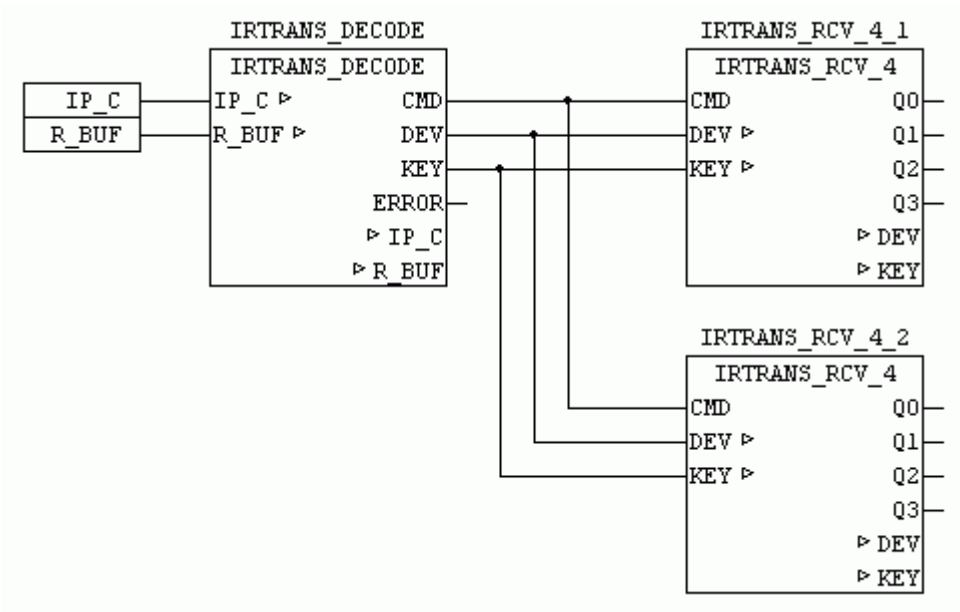
gang DEV dem DEV_CODE (Device Code) entspricht und die Zeichenkette am Eingang KEY dem KEY_CODE entspricht. Wenn die Codes übereinstimmen und CMD = TRUE ist dann wird der Ausgang Q für einen Zyklus auf TRUE gesetzt.

Das folgende Beispiel zeigt die Anwendung von IRTRANS_RCV_1:



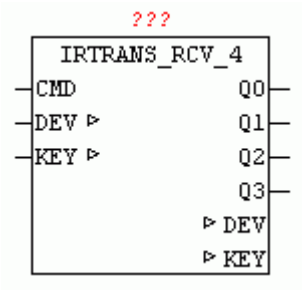
In diesem Beispiel wird der Empfangs-Datenpuffers an IRTRANS_DECODE übergeben. Der Decoder ermittelt aus gültigen Datenpaketen den String DEV und KEY und reicht diese per CMD an IRTRANS_RCV_1 weiter. IRTRANS_RCV_1 oder alternativ auch IRTRANS_RCV_4 und IRTRANS_RCV_8 überprüft dann ob DEV und KEY übereinstimmen und schaltet dann den Ausgang Q für einen Zyklus auf TRUE. im Beispiel wird damit ein DRIVER_1 gesteuert der es der Fernsteuerung erlaubt mit jedem empfangenen Protokoll den Ausgang umzuschalten.

Wenn mehrere Key Codes ausgewertet werden sollen können alternativ die Bausteine IRTRANS_RCV_4 oder IRTRANS_RCV_8 verwendet werden oder mehrere dieser Bausteine parallel betrieben werden.



6.4. IRTRANS_RCV_4

Type	Funktionsbaustein
Input	CMD : BOOL (TRUE wenn Daten zum Auswerten Anliegen)
I/O	DEV : STRING (Name der Fernsteuerung)
	KEY : STRING (NAME der Taste)
Setup	DEV_CODE : STRING (zu dekodierender Fernsteuerungsname)
	KEY_CODE_0..3 : STRING (zu dekodierender Tastencode)
Output	Q0..Q3 : BOOL (Ausgangssignal)

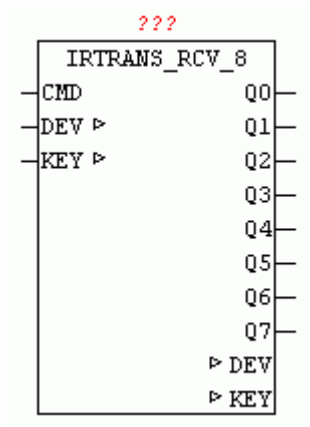


IRTRANS_RCV_4 überprüft wenn CMD = TRUE ob die Zeichenkette am Eingang DEV dem DEV_CODE (Device Code) entspricht und die Zeichenkette am Eingang KEY einem KEY_CODE entspricht. Wenn die Codes übereinstimmen und CMD = TRUE ist dann wird der entsprechende Ausgang Q für einen Zyklus auf TRUE gesetzt. Weitergehende Informationen zur Funktion des Bausteins befinden sich unter IRTRANS_RCV_1.

6.5. IRTRANS_RCV_8

Type	Funktionsbaustein
Input	CMD : BOOL (TRUE wenn Daten zum Auswerten Anliegen)
I/O	DEV : STRING (Name der Fernsteuerung)
	KEY : STRING (NAME der Taste)
Setup	DEV_CODE : STRING (zu dekodierender Fernsteuerungsname)
	KEY_CODE_0..7 : STRING (zu dekodierender Tastencode)
Output	Q0..Q7 : BOOL (Ausgangssignal)

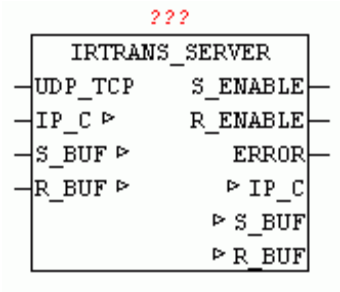
IRTRANS_RCV_8 überprüft wenn CMD = TRUE ob die Zeichenkette am Eingang DEV dem DEV_CODE (Device Code) entspricht und die Zeichenkette am Eingang KEY einem KEY_CODE entspricht. Wenn die Codes überein-



6.6. IRTRANS_SERVER

Sind mehrere CMD = TRUE, so kann wird der entsprechende Ausgang Q für einen Zyklus auf TRUE gesetzt. Weitergehende Informationen zur Funktion des Bausteins befinden sich unter IRTRANS_RCV_1.

Type	Funktionsbaustein
Input	UDP_TCP : BOOL (FALSE = UDP / TRUE = TCP)
In_Out	IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)
	S_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT' (Sendedaten)
	R_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT' (Empfangsdaten)
Output	S_ENABLE : BOOL (IRTRANS Datensenden freigegeben)
	R_ENABLE : BOOL (IRTRANS Datenempfang freigegeben)
	ERROR : DWORD (Fehlercode: Siehe IP_CONTROL)



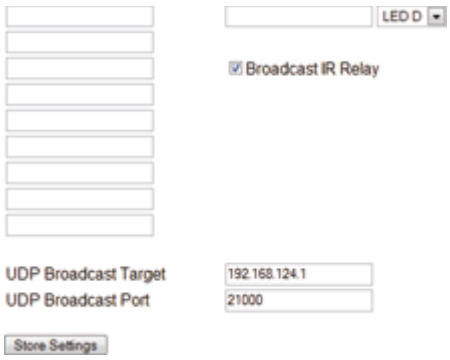
IRTRANS_SERVER kann als sowohl als Empfänger als auch als Sender von IRTRANS-Kommandos benutzt werden. Ist UDP_TCP = TRUE wird eine passive TCP-Verbindung, ansonsten eine passive UDP Verbindung eingerichtet. Die jeweilige Betriebsart muss auch beim IRTRANS Gerät parametriert werden. Sobald eine Datenverbindung vorhanden ist, und das Senden von Kommandos erlaubt ist, wird S_ENABLE = TRUE. Im UDP-Modus können erst nach dem erstmaligen Datenempfang vom IRTRANS, auch Daten gesendet werden, da durch den passiv-UDP-Mode die Partner-IP anfangs noch nicht bekannt ist. Die Empfangsbereitschaft wird mit R_ENABLE angezeigt. Werden Daten empfangen stehen diese im R_BUF zur Weiterver-

arbeitung für andere Bausteine zur Verfügung. Sowie müssen Sendedaten von den Bausteinen im S_BUF eingetragen werden, damit diese dann automatisch vom IRTRANS_SERVER versendet werden. Sollten bei Übertragung Fehler auftreten, so werden diese bei „ERROR“ ausgegeben (Siehe Baustein IP_CONTROL2). Vorhandene Fehler werden automatisch alle 5 Sekunden vom Baustein quittiert.

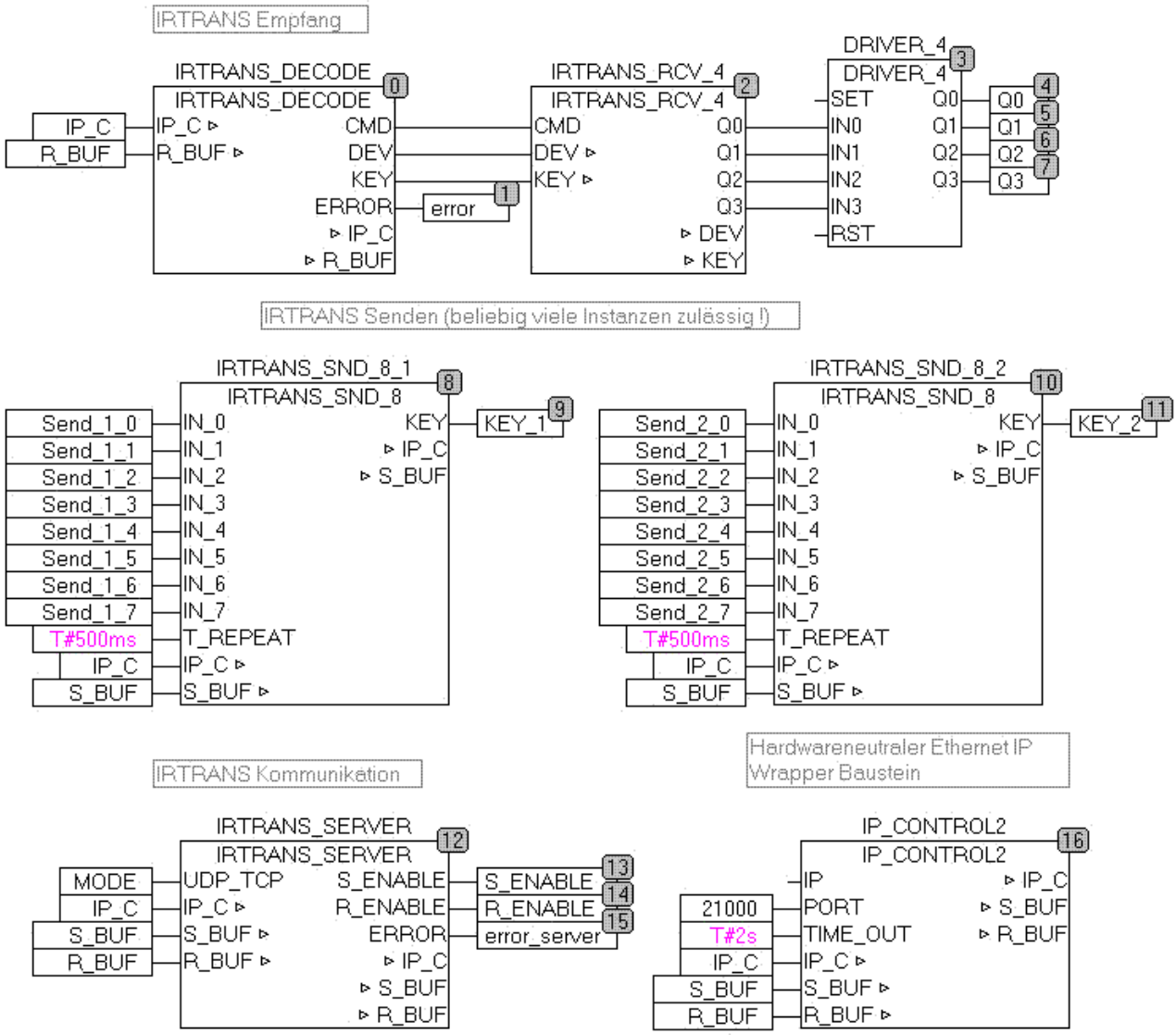
UDP-SERVER Modus:

In der IRTRANS Web Konfiguration muss als Broadcast Adresse die IP Adresse der SPS eingetragen werden.

IRTRANS Web Konfiguration:



Das folgende Beispiel zeigt die Anwendung von IRTRANS Bausteine

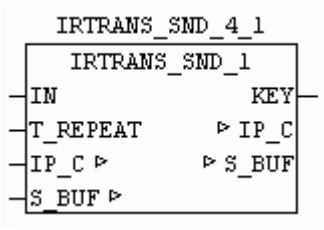


6.7. IRTRANS_SND_1

Type	Funktionsbaustein
Input	IN : BOOL (TRUE = Keycode senden)
	T_REPEAT : TIME (Zeit für erneutes Senden des Tastencodes)
I/O	IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)
	S_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT' (Sendedaten)
Setup	DEV_CODE : STRING (zu sendender Fernsteuerungsname)

KEY_CODE : STRING (zu sendender Tastencode)

Output KEY : BYTE (Ausgabe des gerade aktiven Keycodes)



IRTRANS_SND_1 ermöglicht das Senden eines Fernsteuerbefehles an den IRTrans. Wenn IN TRUE ist wird der im Setup angegebene Device und Keycode an den IRTrans versendet, der diese wiederum als echte Fernsteuerbefehle ausgibt. Mittels T_REPEAT kann die Wiederholungszeit für das Senden vorgeben werden. Bleibt IN ständig auf TRUE so wird immer nach Ablauf der Zeit T_REPEAT wiederholt dieser Tastencode versendet. Beim Output KEY wird bei aktiver Ansteuerung „1“ ausgegeben. KEY = 0 bedeutet das IN nicht aktiv ist.

6.8. IRTRANS_SND_4

Type Funktionsbaustein

Input IN_0..3 : BOOL (TRUE = Keycode x senden)

T_REPEAT : TIME (Zeit für erneutes Senden des Tastencodes)

I/O IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)

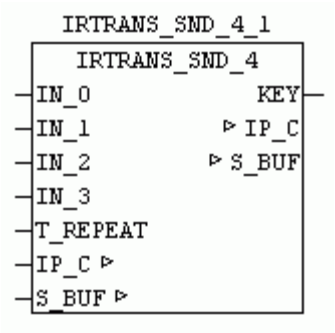
S_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT'

(Sendedaten)

Setup DEV_CODE : STRING (zu sendender Fernsteuerungsname)

KEY_CODE_0..3 : STRING (zu sendender Tastencode)

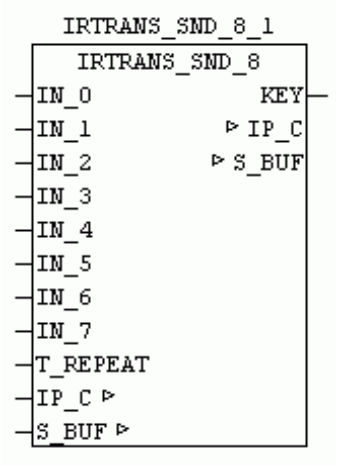
Output KEY : BYTE (Ausgabe des gerade aktiven Keycodes)



IRTRANS_SND_4 ermöglicht das Senden von Fernsteuerbefehlen an den IR-Trans. Wenn IN_x TRUE ist wird der im Setup angegebene Device und Keycode an den IRTrans versendet, der diese wiederum als echte Fernsteuerbefehle ausgibt. Mittels T_REPEAT kann die Wiederholungszeit für das Senden vorgeben werden. Bleibt z.B. IN_0 ständig auf TRUE so wird immer nach Ablauf der Zeit T_REPEAT wiederholt dieser Tastencode versendet. Erfolgt ein Wechsel auf einen anderen IN_x so wird dieser Code sofort versendet, und dann erst wieder über T_REPEAT verzögert , sollte dieser längere Zeit anstehen. Beim Output KEY wird der gerade aktuell angesteuerte KEY angezeigt. KEY = 0 bedeutet das kein IN_x aktiv ist. Die Werte 1-3 entsprechen den IN0 – IN_3.

6.9. IRTRANS_SND_8

Type	Funktionsbaustein
Input	IN_0..7 : BOOL (TRUE = Keycode x senden)
	T_REPEAT : TIME (Zeit für erneutes Senden des Tastencodes)
I/O	IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)
	S_BUF: Datenstruktur 'NETWORK_BUFFER_SHORT'
	(Sendedaten)
Setup	DEV_CODE : STRING (zu sendender Fernsteuerungsname)
	KEY_CODE_0..7 : STRING (zu sendender Tastencode)
Output	KEY : BYTE (Ausgabe des gerade aktiven Keycodes)



IRTRANS_SND_8 ermöglicht das Senden von Fernsteuerbefehlen an den IR-Trans. Wenn IN_x TRUE ist wird der im Setup angegebene Device und Key-code an den IRTrans versendet, der diese wiederum als echte Fernsteuer-befehle ausgibt. Mittels T_REPEAT kann die Wiederholungszeit für das Sen-den vorgeben werden. Bleibt z.B. IN_0 ständig auf TRUE so wird immer nach Ablauf der Zeit T_REPEAT wiederholt dieser Tastencode versendet. Erfolgt ein Wechsel auf einen anderen IN_x so wird dieser Code sofort ver-sendet, und dann erst wieder über T_REPEAT verzögert , sollte dieser län-gere Zeit anstehen. Beim Output KEY wird der gerade aktuell angesteuer-te KEY angezeigt. KEY = 0 bedeutet das kein IN_x aktiv ist. Die Werte 1-8 entsprechen den IN0 – IN_7.

7. Daten Logger

7.1. DATEN-LOGGER

Die Daten-Logger Bausteine ermöglichen das Sammeln und Speichern von Prozessdaten in Echtzeit. Nach Auslösen des Speicherimpulses werden alle parametrierten Prozesswerte in einen Datenbuffer zwischengespeichert, da die verschiedenen Speichermedien meistens nicht schnell genug sind. Es können bis zu 255 Prozesswerte in einem Paket verarbeitet werden. Die Aufrufreihenfolge der Bausteine bestimmt automatisch die Reihung der Prozesswerte (Datenfluss-Reihenfolge beachten !)

Zur Speicherung der verschiedenen Datentypen werden folgende Bausteine zur Verfügung gestellt.

DLOG_STRING

DLOG_REAL

DLOG_DINT

DLOG_DT

DLOG_BOOL

Sonstige Datentypen vorher manuell konvertieren, und als STRING übergeben. Die gesammelten Daten können dann auf ein Datenziel weitergeleitet werden.

DLOG_STORE_FILE_CSV (Daten als CSV-Datei speichern)

DLOG_STORE_FILE_HTML (Daten als HTML-Datei speichern

DLOG_STORE_FILE_XML (Daten als XML-Datei speichern)

DLOG_STORE_RRD (Daten auf RRD-Server speichern)

DLOG_STORE_MYSQL (Daten auf MYSQL-Server speichern)

Die Dateien die auf der Steuerung abgelegt werden, können dann auf externe Datenziele weitergeleitet werden.

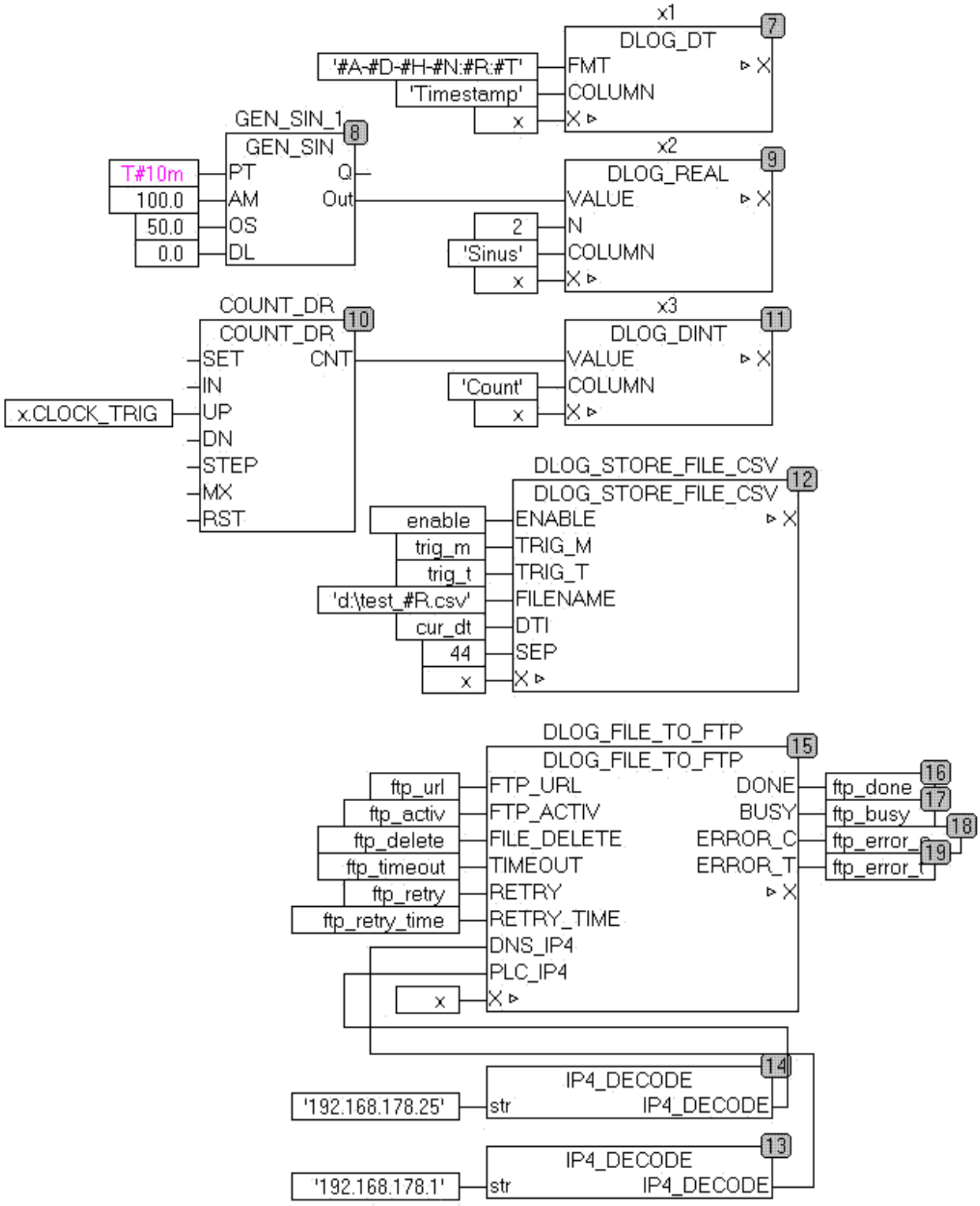
DLOG_FILE_TO_SMTP (Datei als Email versenden)

DLOG_FILE_TO_FTP (Datei auf einen externen FTP-Server kopieren)

Die oben angeführten Module können folgend miteinander kombiniert werden.

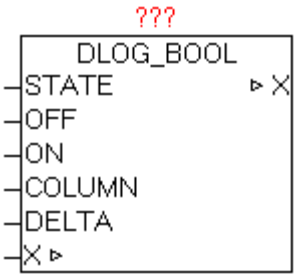
	DLOG_BOOL	DLOG_DINT	DLOG_DT	DLOG_REAL	DLOG_STRING	DLOG_FILE_TO_FTP	DLOG_FILE_TO_SMTP
DLOG_STORE_FILE_CSV	x	x	x	x	x	x	x
DLOG_STORE_FILE_HTML	x	x	x	x	x	x	x
DLOG_STORE_FILE_XML	x	x	x	x	x	x	x
DLOG_STORE_RRD	x	x	x	x	x		
DLOG_STORE_MYSQL	x	x	x	x	x		

Nachfolgendes Beispiel zeigt die Aufzeichnung von einen Zeitstempel, einem REAL und DINT Zähler. Dabei werden die Prozessdaten nach jeder Minute in einer neuen CSV formatierten Datei abgelegt. Sobald eine Datei fertig ist, wird diese automatisch auf einen FTP-Server verschoben.



7.2. DLOG_BOOL

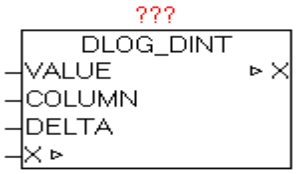
Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	STATE : BOOL (Prozesswert TRUE/FALSE)
	ON : STRING (Text für Zustand TRUE)
	OFF : STRING (Text für Zustand FALSE)
	COLUMN : STRING(40) (Prozesswertbezeichnung)
	DELTA : DINT (Differenzwert)



Der Baustein DLOG_BOOL dient zum Loggen (Aufzeichnen) eines Prozesswertes vom Typ BOOL, und kann nur in Kombination mit einem DLOG_STORE_* Baustein verwendet werden, da dieser über die Datenstruktur X die Aufzeichnung der Daten koordiniert. Bei Aufzeichnungsformaten die eine Prozesswertbezeichnung unterstützen wie z.B. DLOG_STORE_FILE_CSV kann bei „COLUMN“ ein Name vorgeben werden. Abhängig vom Zustand von STATE wird der TEXT von Parameter OFF oder ON verwendet. Wird bei Parameter DELTA ein TRUE vorgegeben, wird das automatischen Aufzeichnen über Differenzüberwachung aktiviert. Verändert sich der Zustand von STATE wird automatisch ein Datensatz gespeichert. Diese Funktion kann parallel zu den zentralen Trigger über die DLOG_STORE_* Bausteine angewendet werden.

7.3. DLOG_DINT

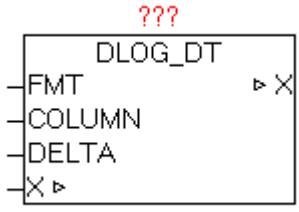
Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	VALUE : DINT (Prozesswert)
	COLUMN : STRING(40) (Prozesswertbezeichnung)
	DELTA : DINT (Differenzwert)



Der Baustein DLOG_DINT dient zum Loggen (Aufzeichnen) eines Prozesswertes vom Typ DINT, und kann nur in Kombination mit einem DLOG_STORE_* Baustein verwendet werden, da dieser über die Datenstruktur X die Aufzeichnung der Daten koordiniert. Bei Aufzeichnungsformaten die eine Prozesswertbezeichnung unterstützen wie z.B. DLOG_STORE_FILE_CSV kann bei „COLUMN“ ein Name vorgeben werden. Wird bei Parameter DELTA ein Wert ungleich 0 vorgegeben, wird das automatischen Datenloggen über Differenzüberwachung aktiviert. Verändert sich der Wert von VALUE um +/- DELTA wird automatisch ein Datensatz gespeichert. Diese Funktion kann parallel zu den zentralen Trigger über die DLOG_STORE_* Bausteine angewendet werden.

7.4. DLOG_DT

Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	FMT : STRING (Formatierungsparameter)
	COLUMN : STRING(40) (Prozesswertbezeichnung)
	DELTA : UDINT (Differenzwert in Sekunden)



Der Baustein DLOG_DT dient zum Loggen (Aufzeichnen) eines Datum oder Zeit Wertes vom Typ STRING, und kann nur in Kombination mit einem DLOG_STORE_* Baustein verwendet werden, da dieser über die Datenstruktur X die Aufzeichnung der Daten koordiniert. Mittels Parameter FMT kann die Formatierung vorgegeben werden werden. Im Parameter FMT kann auch normaler Text mit Formatierungsparametern kombiniert werden. Siehe Dokumentation vom Baustein DT_TO_STRF. Wird der

Parameter FMT nicht angegeben, so wird die Standardformatierung '#A-#D-#H #N:#R:#T' verwendet

Bei Aufzeichnungsformaten die eine Prozesswertbezeichnung unterstützen wie z.B. DLOG_STORE_FILE_CSV kann bei „COLUMN“ ein Name vorgeben werden.

Wird bei Parameter DELTA ein Wert größer 0 vorgegeben, wird das automatischen Datenloggen über Differenzüberwachung aktiviert. Verändert sich die Zeit um den Wert von DELTA wird automatisch ein Datensatz gespeichert. Diese Funktion kann parallel zu den zentralen Trigger über die DLOG_STORE_* Bausteine angewendet werden. Wird z.B. Bei DELTA der Wert 30 angeben, wird automatisch alle 30 Sekunden ein Datensatz gespeichert.

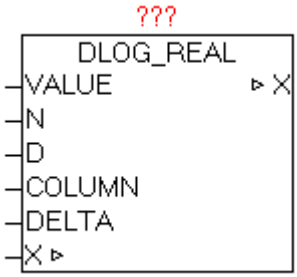
Beispiel:

FMT := '#A-#D-#H-#N:#R:#T'

ergibt '2011-12-22-06:12:50'

7.5. DLOG_REAL

Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	VALUE : REAL (Prozesswert)
	N : INT (Anzahl der Nachkommastellen)
	D : STRING(1) (Dezimalpunktzeichen)
	COLUMN : STRING(40) (Prozesswertbezeichnung)
	DELTA : REAL (Differenzwert)

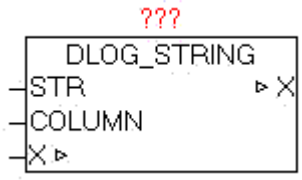


Der Baustein DLOG_REAL dient zum Loggen (Aufzeichnen) eines Prozesswertes vom Typ REAL, und kann nur in Kombination mit einem DLOG_STORE_* Baustein verwendet werden, da dieser über die Datenstruktur X die Aufzeichnung der Daten koordiniert. Mittels Parameter

N kann die Anzahl der gewünschten Nachkommastellen vorgeben werden. Siehe Dokumentation vom Baustein REAL_TO_STRF. Der Eingang D legt fest mit welchem Zeichen der Dezimalpunkt dargestellt wird. Wir bei Parameter D kein Zeichen übergeben, so wird automatisch ',' verwendet. Bei Aufzeichnungsformaten die eine Prozesswertbezeichnung unterstützen wie z.B. DLOG_STORE_FILE_CSV kann bei „COLUMN“ ein Name vorgeben werden. Wird bei Parameter DELTA ein Wert ungleich 0.0 vorgegeben, wird das automatischen Datenloggen über Differenzüberwachung aktiviert. Verändert sich der Wert von VALUE um +/- DELTA wird automatisch ein Datensatz gespeichert. Diese Funktion kann parallel zu den zentralen Trigger über die DLOG_STORE_* Bausteine angewendet werden.

7.6. DLOG_STRING

Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	STR : STRING (Prozesswert)
	COLUMN : STRING(40) (Prozesswertbezeichnung)

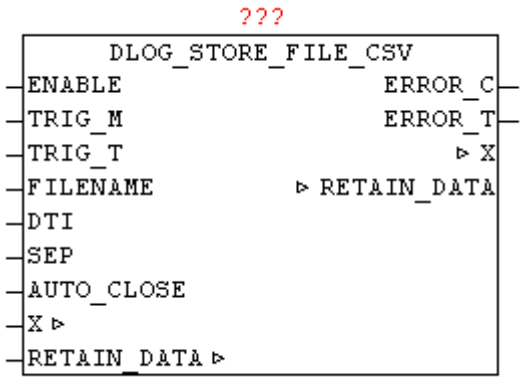


Der Baustein DLOG_STRING dient zum Loggen (Aufzeichnen) eines Prozesswertes vom Typ STRING, und kann nur in Kombination mit einem DLOG_STORE_* Baustein verwendet werden, da dieser über die Datenstruktur X die Aufzeichnung der Daten koordiniert. Bei Aufzeichnungsformaten die eine Prozesswertbezeichnung unterstützen wie z.B. DLOG_STORE_FILE_CSV kann bei „COLUMN“ ein Name vorgeben werden.

7.7. DLOG_STORE_FILE_CSV

Type	Funktionsbaustein:
INPUT	ENABLE : BOOL (Datenaufzeichnung freigegen)
	TRIG_M : BOOL (Manueller Auslöser)
	TRIG_T : UDINT (Automatischer Auslöser über Zeit)

	FILENAME : STRING (Dateiname)
	DTI : DT (Aktueller DATE-TIME Wert)
	SEP : BYTE (Trennzeichen der aufgezeichneten Elemente)
	AUTO_CLOSE : TIME (Zeit für automatisches Datei schließen)
OUTPUT	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
	RETAIN_DATA : DLOG_RETAIN (Remanente Daten)



Der Baustein DLOG_STORE_FILE_CSV dient zum Loggen (Aufzeichnen) der Prozesswerte in eine CSV formatierte Datei. Die Daten können mit den Bausteinen DLOG_DINT, DLOG_REAL, DLOG_STRING, DLOG_DT übergeben werden. Der Parameter TRIG_M (positiver Impuls) dient zum manuellen Triggern (Auslösen) der Speicherung der Prozessdaten. Mittels Parameter TRIG_T kann eine automatische zeitgesteuerte Auslösung realisiert werden. Wenn der aktuelle Datum/Zeit Wert durch den parametrierten TRIG_T Wert mit Rest 0 teilbar ist, dann wird eine Speicherung ausgelöst. Dadurch ist auch sichergestellt das die Speicherung immer zum gleichen Zeitpunkt durchgeführt wird.

Beispiele:

TRIG_T = 60

alle 60 Sekunden also bei jeder neuen Minute in Sekunde 0 wird gespeichert.

TRIG_T = 10

Bei Sekunde 0,10,20,30,40,50 wird eine Speicherung ausgelöst

TRIG_T = 3600

nach jeder neuen Stunde also bei Minute 0 und Sekunde 0 wird gespeichert.

Die Auslöser TRIG_T und TRIG_M können parallel von einander unabhängig benutzt werden.

Mittels Parameter FILENAME wird der Dateiname (inklusive Dateipfad wenn notwendig) vorgegeben. Wird der Dateiname während der Auszeichnung geändert, so wird automatisch auf die neuen Aufzeichnungsdatei fliegend (ohne Datenverlust) gewechselt. Dieser Wechsel kann auch automatisiert werden. Der Parameter FILENAME unterstützt die Verwendung von Datum/Zeit Parameter (Siehe Dokumentation vom Baustein DT_TO_STRF)

Beispiel: FILENAME = 'Station_01_#R.csv'

An Position von '#R' wird automatisch die aktuelle Minutenanzahl eingetragen. Das heißt das automatisch nach jeder Minute sich der Dateiname ändert, und dementsprechend werden die Daten in die jeweilige Datei geschrieben. Somit werden innerhalb einer ganzen Stunde 60 Dateien angelegt und mit Daten beschrieben, und auch im Ring-Buffer verfahren immer wieder überschrieben.

Damit lässt sich automatisch nach belieben eine Aufzeichnung durchführen die z.B. jeden Tag , Woche, Monat usw. eine neue Datei erzeugt. Wird ein neuer FILENAME erkannt so wird eine eventuell schon vorhandene Datei gelöscht, und neu beschrieben.

Am Parameter DTI muss die aktuelle Datum/Zeitwert übergeben werden. Bei SEP wird der ASCII-Code des Trennzeichens übergeben.

Mit dem Parameter AUTO_CLOSE kann die maximale Zeit vorgegeben werden, nach der der interne Datenbuffer automatisch in die Datei geschrieben wird und die Datei geschlossen wird.

Die Datenstruktur „RETAIN_DATA“ dient zum möglichst problemlosen fortsetzen der Datenaufzeichnung nach einem Spannungsausfall, sodass die bestehende Datei nicht gelöscht wird, und muss zwingend als VAR_RETAIN (Remanente Variable) deklariert werden.

CSV-Dateiformat:

Siehe: http://de.wikipedia.org/wiki/CSV_(Dateiformat)

Beispiel einer CSV-Datei mit Trennzeichen ';' und Spaltenüberschriften

Date / Time;Z1;Z2;Sekunden

2010-10-22-06:00:00;1;2;00

2010-10-22-06:00:06;1;2;06

2010-10-22-06:00:12;1;2;12

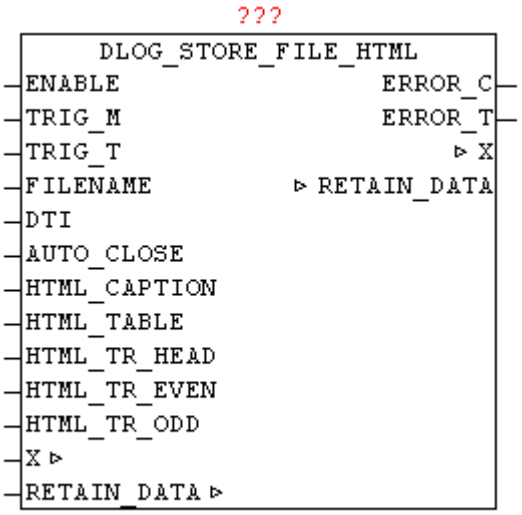
2010-10-22-06:00:18;1;2;18

ERROR_T:

Wert	Eigenschaften
1	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen

7.8. DLOG_STORE_FILE_HTML

Type	Funktionsbaustein:
INPUT	ENABLE : BOOL (Datenaufzeichnung freigeben)
	TRIG_M : BOOL (Manueller Auslöser)
	TRIG_T : UDINT (Automatischer Auslöser über Zeit)
	FILENAME : STRING (Dateiname)
	DTI : DT (Aktueller DATE-TIME Wert)
	AUTO_CLOSE : TIME (Zeit für automatisches Datei schließen)
	HTML_CAPTION : STRING (HTML Code für die Überschrift)
	HTML_TABLE : STRING (HTML Code für die Tabelle)
	HTML_TR_HEAD : STRING (HTML Code der Tabellenüberschrift)
	HTML_TR_EVEN : STRING (HTML Code der ungeraden Zeilen)
	HTML_TR_ODD : STRING (HTML Code der geraden Zeilen)
OUTPUT	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
	RETAIN_DATA : DLOG_RETAIN (Remanente Daten)



Der Baustein DLOG_STORE_FILE_HTML dient zum Loggen (Aufzeichnen) der Prozesswerte in eine HTML Datei in der die Daten als Tabelle dargestellt werden. Mit dem HTML_* Parametern kann an den Schlüsselstellen beliebiger HTML-Code eingefügt werden. Dabei geht es vorwiegend um Formatierungsparameter wie Schriften,Größe und Farbe.

Die Daten können mit den Bausteinen DLOG_DINT, DLOG_REAL, DLOG_STRING, DLOG_DT übergeben werden. Der Parameter TRIG_M (positiver Impuls) dient zum manuellen Triggern (Auslösen) der Speicherung der Prozessdaten. Mittels Parameter TRIG_T kann eine automatische zeitgesteuerte Auslösung realisiert werden. Wenn der aktuelle Datum/Zeit Wert durch den parametrierten TRIG_T Wert mit Rest 0 teilbar ist, dann wird eine Speicherung ausgelöst.
Dadurch ist auch sichergestellt das die Speicherung immer zum gleichen Zeitpunkt durchgeführt wird

Beispiele:

TRIG_T = 60

alle 60 Sekunden also bei jeder neuen Minute in Sekunde 0 wird gespeichert.

TRIG_T = 10

Bei Sekunde 0,10,20,30,40,50 wird eine Speicherung ausgelöst

TRIG_T = 3600

nach jeder neuen Stunde also bei Minute 0 und Sekunde 0 wird gespeichert.

Die Auslöser TRIG_T und TRIG_M können parallel von einander unabhängig benutzt werden.

Mittels Parameter FILENAME wird der Dateiname (inklusive Dateipfad wenn notwendig) vorgegeben. Wird der Dateiname während der Auszeichnung geändert, so wird automatisch auf die neuen Aufzeichnungsdatei fliegend (ohne Datenverlust) gewechselt. Dieser Wechsel kann auch automatisiert werden. Der Parameter FILENAME unterstützt die Verwendung von Datum/Zeit Parameter (Siehe Dokumentation vom Baustein DT_TO_STRF)

Beispiel: FILENAME = 'Station_01_#R.html'

An Position von '#R' wird automatisch die aktuelle Minutenanzahl eingetragen. Das heißt das automatisch nach jeder Minute sich der Dateiname ändert, und dementsprechend werden die Daten in die jeweilige Datei geschrieben. Somit werden innerhalb einer ganzen Stunde 60 Dateien angelegt und mit Daten beschrieben, und auch im Ring-Buffer verfahren immer wieder überschrieben.

Damit lässt sich automatisch nach belieben eine Aufzeichnung durchführen die z.B. jeden Tag , Woche, Monat usw. eine neue Datei erzeugt. Wird ein neuer FILENAME erkannt so wird eine eventuell schon vorhandene Datei gelöscht, und neu beschrieben.

Am Parameter DTI muss die aktuelle Datum/Zeitwert übergeben werden. Bei SEP wird der ASCII-Code des Trennzeichens übergeben.

Mit dem Parameter AUTO_CLOSE kann die maximale Zeit vorgegeben werden, nach der der interne Datenbuffer automatisch in die Datei geschrieben wird und die Datei geschlossen wird.

Die Datenstruktur „RETAIN_DATA“ dient zum möglichst problemlosen fortsetzen der Datenaufzeichnung nach einem Spannungsausfall, sodass die bestehende Datei nicht gelöscht wird, und muss zwingend als VAR_RETAIN (Remanente Variable) deklariert werden.

ERROR_T:

Wert	Eigenschaften
1	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen

HTML-Dateiformat:

http://de.wikipedia.org/wiki/Html

HTML-Farbcodes

http://html-color-codes.info/webfarben_hexcodes/

http://www.uni-magdeburg.de/counter/rgb.txt.shtml

Beispiel einer HTML-Datei die mit dem Demoprogramm DLOG_FILE_HTML_DEMO erzeugt wurde. Als HTML-Parameter wurden folgende Vorgaben benutzt

html_caption: STRING := 'Das ist der Titel';

html_tr_even: STRING := 'BGCOLOR=#B3B7FF';

html_tr_odd: STRING := 'BGCOLOR=#E0E0E0';

html_tr_head: STRING := 'BGCOLOR=#FFFF40';

html_table : STRING := 'BORDER="1"';

Das ist der Titel

Timestamp	Sinus	Count	Count_Bit_2	Count_Hex
2010-10-22 06:00:00	50.01	1	OFF	00000000000000000000000000000001
2010-10-22 06:00:05	52.62	5	ON	00000000000000000000000000000101
2010-10-22 06:00:10	55.23	10	OFF	00000000000000000000000000001010
2010-10-22 06:00:15	57.82	15	ON	00000000000000000000000000001111
2010-10-22 06:00:20	60.40	20	ON	00000000000000000000000000010100
2010-10-22 06:00:25	62.94	25	OFF	00000000000000000000000000011001
2010-10-22 06:00:30	65.45	30	ON	00000000000000000000000000011110
2010-10-22 06:00:35	67.92	35	OFF	00000000000000000000000000100011
2010-10-22 06:00:40	70.34	40	OFF	00000000000000000000000000101000
2010-10-22 06:00:45	72.70	45	ON	00000000000000000000000000101101
2010-10-22 06:00:50	75.00	50	OFF	00000000000000000000000000110010
2010-10-22 06:00:55	77.23	55	ON	00000000000000000000000000110111

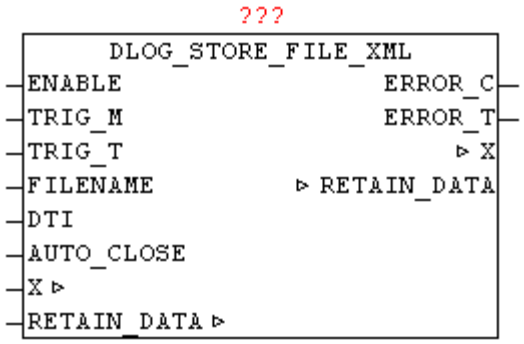
Die erzeugten HTML Daten sehen im Text-Editor folgende aus.

```
<html>
 <body>
  <table BORDER="1">
   <caption>Das ist der Titel
   </caption>
   <TR BGCOLOR=#FFFF40>
    <TD>Timestamp</TD>
    <TD>Sinus</TD>
    <TD>Count</TD>
```

```
      <TD>Count_Bit_2</TD>
      <TD>Count_Hex</TD>
    </TR>
    <TR BGCOLOR=#B3B7FF>
      <TD>2010-10-22-06:00:00</TD>
      <TD>50.01</TD>
      <TD>1</TD>
      <TD>OFF</TD>
      <TD>00000000000000000000000000000001</TD>
    </TR>
    ……..
    ……..
  </table>
 </body>
</html>
```

7.9. DLOG_STORE_FILE_XML

Type	Funktionsbaustein:
INPUT	ENABLE : BOOL (Datenaufzeichnung freigeben)
	TRIG_M : BOOL (Manueller Auslöser)
	TRIG_T : UDINT (Automatischer Auslöser über Zeit)
	FILENAME : STRING (Dateiname)
	DTI : DT (Aktueller DATE-TIME Wert)
	AUTO_CLOSE : TIME (Zeit für automatisches Datei schließen)
OUTPUT	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
	RETAIN_DATA : DLOG_RETAIN (Remanente Daten)



Der Baustein DLOG_STORE_FILE_XML dient zum Loggen (Aufzeichnen) der Prozesswerte in eine XML Datei.Die Daten können mit den Bausteinen DLOG_DINT, DLOG_REAL, DLOG_STRING, DLOG_DT übergeben werden. Der Parameter TRIG_M (positiver Impuls) dient zum manuellen Triggern (Auslösen) der Speicherung der Prozessdaten. Mittels Parameter TRIG_T kann eine automatische zeitgesteuerte Auslösung realisiert werden. Wenn der aktuelle Datum/Zeit Wert durch den parametrierten TRIG_T Wert mit Rest 0 teilbar ist, dann wird eine Speicherung ausgelöst.
Dadurch ist auch sichergestellt das die Speicherung immer zum gleichen Zeitpunkt durchgeführt wird

Beispiele:

TRIG_T = 60

alle 60 Sekunden also bei jeder neuen Minute in Sekunde 0 wird gespeichert.

TRIG_T = 10

Bei Sekunde 0,10,20,30,40,50 wird eine Speicherung ausgelöst

TRIG_T = 3600

nach jeder neuen Stunde also bei Minute 0 und Sekunde 0 wird gespeichert.

Die Auslöser TRIG_T und TRIG_M können parallel von einander unabhängig benutzt werden.

Mittels Parameter FILENAME wird der Dateiname (inklusive Dateipfad wenn notwendig) vorgegeben. Wird der Dateiname während der Auszeichnung geändert, so wird automatisch auf die neuen Aufzeichnungsdatei fliegend (ohne Datenverlust) gewechselt. Dieser Wechsel kann auch automatisiert werden. Der Parameter FILENAME unterstützt die Verwendung von Datum/Zeit Parameter (Siehe Dokumentation vom Baustein DT_TO_STRF)

Beispiel: FILENAME = 'Station_01_#R.xml'

An Position von '#R' wird automatisch die aktuelle Minutenanzahl eingetragen. Das heißt das automatisch nach jeder Minute sich der Dateiname ändert, und dementsprechend werden die Daten in die jeweilige Datei geschrieben. Somit werden innerhalb einer ganzen Stunde 60 Dateien angelegt und mit Daten beschrieben, und auch im Ring-Buffer verfahren immer wieder überschrieben.

Damit lässt sich automatisch nach belieben eine Aufzeichnung durchführen die z.B. jeden Tag , Woche, Monat usw. eine neue Datei erzeugt. Wird ein neuer FILENAME erkannt so wird eine eventuell schon vorhandene Datei gelöscht, und neu beschrieben.

Am Parameter DTI muss die aktuelle Datum/Zeitwert übergeben werden. Bei SEP wird der ASCII-Code des Trennzeichens übergeben.

Mit dem Parameter AUTO_CLOSE kann die maximale Zeit vorgegeben werden, nach der der interne Datenbuffer automatisch in die Datei geschrieben wird und die Datei geschlossen wird.

Die Datenstruktur „RETAIN_DATA“ dient zum möglichst problemlosen fortsetzen der Datenaufzeichnung nach einem Spannungsausfall, sodass die bestehende Datei nicht gelöscht wird, und muss zwingend als VAR_RETAIN (Remanente Variable) deklariert werden.

ERROR_T:

Wert	Eigenschaften
1	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen

XML-Dateiformat: http://de.wikipedia.org/wiki/Xml

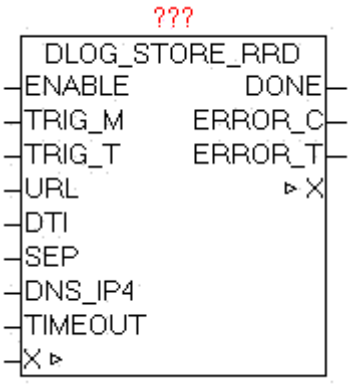
Beispiel einer XML-Datei die mit dem Demoprogramm DLOG_FILE_XML_DEMO erzeugt wurde.

Die Daten sind innerhalb des Elements <table> angelegt. Die Datensätze selber sind innerhalb des <row> Elemente dargestellt. Wobei das erste <row> Element die Prozesswert-Namen beinhaltet. Alle nachfolgenden <row> Elemente stellen direkt die Prozesswerte dar.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <table>
  - <row>
      <entry>Timestamp</entry>
      <entry>Sinus</entry>
      <entry>Count</entry>
      <entry>Count_Bit_2</entry>
      <entry>Count_Hex</entry>
    </row>
  - <row>
      <entry>2010-10-22 06:01:00</entry>
      <entry>79.39</entry>
      <entry>61</entry>
      <entry>ON</entry>
      <entry>00000000000000000000000000111101</entry>
    </row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
  + <row>
 </table>
```

7.10. DLOG_STORE_RRD

Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	ENABLE : BOOL (Datenaufzeichnung freigeben)
	TRIG_M : BOOL (Manueller Auslöser)
	TRIG_T : UDINT (Automatischer Auslöser über Zeit)
	URL : STRING(STRING_LENGTH) (URL-Adresse des Servers)
	DTI : DT (Aktueller DATE-TIME Wert)
	SEP : BYTE (Trennzeichen der aufgezeichneten Elemente)
	DNS_IP4 : DWORD (IP-Adresse des DNS-Servers)
	TIMEOUT : TIME (Überwachungszeit)
OUTPUT	DONE : BOOL (Datenübertragung ohne Fehler beendet)
	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)



Der Baustein DLOG_STORE_RRD dient zum Loggen (Aufzeichnen) der Prozesswerte in eine RRD Datenbank. Die Daten können mit den Bausteinen DLOG_DINT, DLOG_REAL, DLOG_STRING, DLOG_DT übergeben werden. Der Parameter TRIG_M (positiver Impuls) dient zum manuellen Triggern (Auslösen) der Speicherung der Prozessdaten. Mittels Parameter TRIG_T kann eine automatische zeitgesteuerte Auslösung realisiert werden. Wenn der aktuelle Datum/Zeit Wert durch den parametrierten TRIG_T Wert mit Rest 0 teilbar ist, dann wird eine Speicherung ausgelöst. Dadurch ist auch sichergestellt das die Speicherung immer zum gleichen Zeitpunkt durchgeführt wird

Beispiele:

TRIG_T = 60

alle 60 Sekunden, also bei jeder neuen Minute in Sekunde 0 wird gespeichert.

TRIG_T = 10

Bei Sekunde 0,10,20,30,40,50 wird eine Speicherung ausgelöst

TRIG_T = 3600

nach jeder neuen Stunde, also bei Minute 0 und Sekunde 0 wird gespeichert.

Die Auslöser TRIG_T und TRIG_M können parallel von einander unabhängig benutzt werden.

Am Parameter DTI muss die aktuelle Datum/Zeitwert übergeben werden. Bei SEP wird der ASCII-Code des Trennzeichens übergeben.

Sollte dabei ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.

ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen

2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen
3	ERROR_C = 1: Die Daten wurden vom RRD-Server (PHP-Script) nicht übernommen
4	ERROR_C = 1: Die Daten konnten nicht im URL-String übergeben werden. Anzahl der Parameter bzw. Menge der Daten reduzieren (URL+Daten <= 250 Zeichen)

Mit dem Parameter URL wird der Zugriffspfad, und der php-script Aufruf übergeben.

Beispiel einer URL:

http://my_servername/myhouse/rrd/test_rrd.php?rrd_db=test.rrd&value=

DNS-Server oder IP-Adresse

Zugriffspfad und Name des php-script

php-script parameter 1 = Datenbankname

php-script parameter 2 = Prozesswerte

Der Baustein kopiert automatisch alle Prozesswerte hinter „&value=“

sodass sich dann folgende Daten (Exemplarisch) verwendet werden

http://my_servername/myhouse/rrd/test_rrd.php?
rrd_db=test.rrd&value=10:20:30:40:50:60:70

Die einzelnen Prozessdaten werden mittels dem Parameter SEP (Trennzeichen) voneinander abgetrennt.

Es ist zu beachten das der übergebene URL-String und die Prozessdaten nicht mehr als 250 Zeichen erreichen.

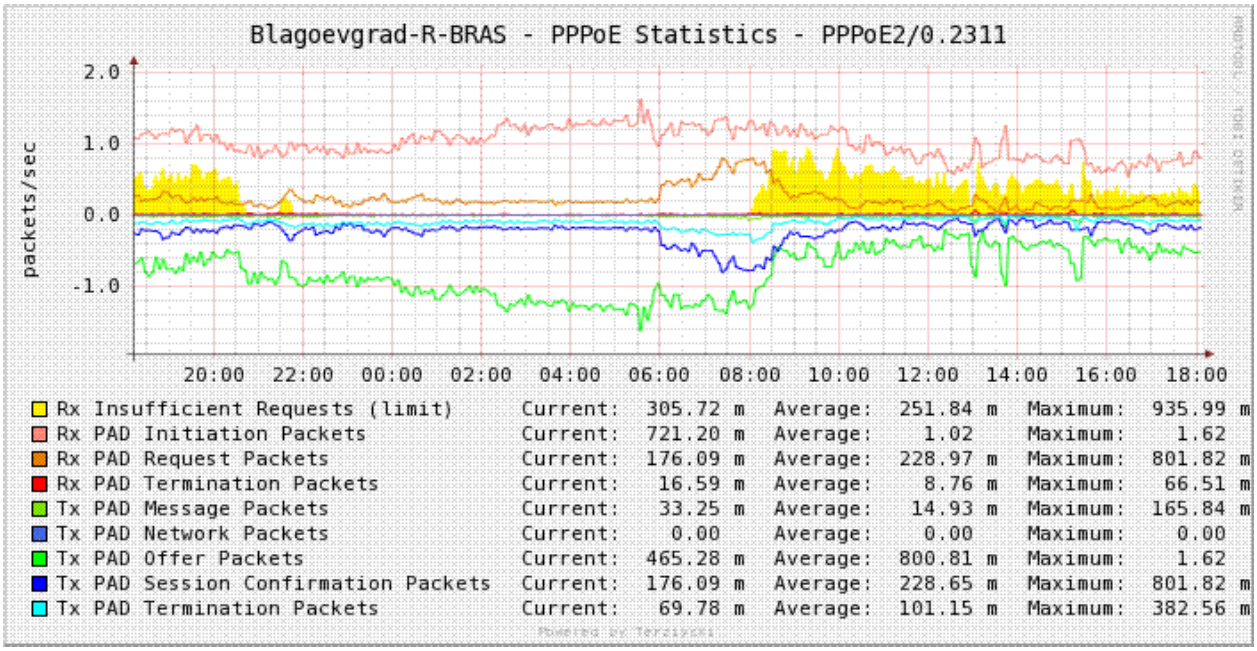
Der Aufbau der URL ist nur Beispielhaft, und kann in Verbindung mit eigenen Server-Applikationen und Scripten im Prinzip frei gestaltet werden.

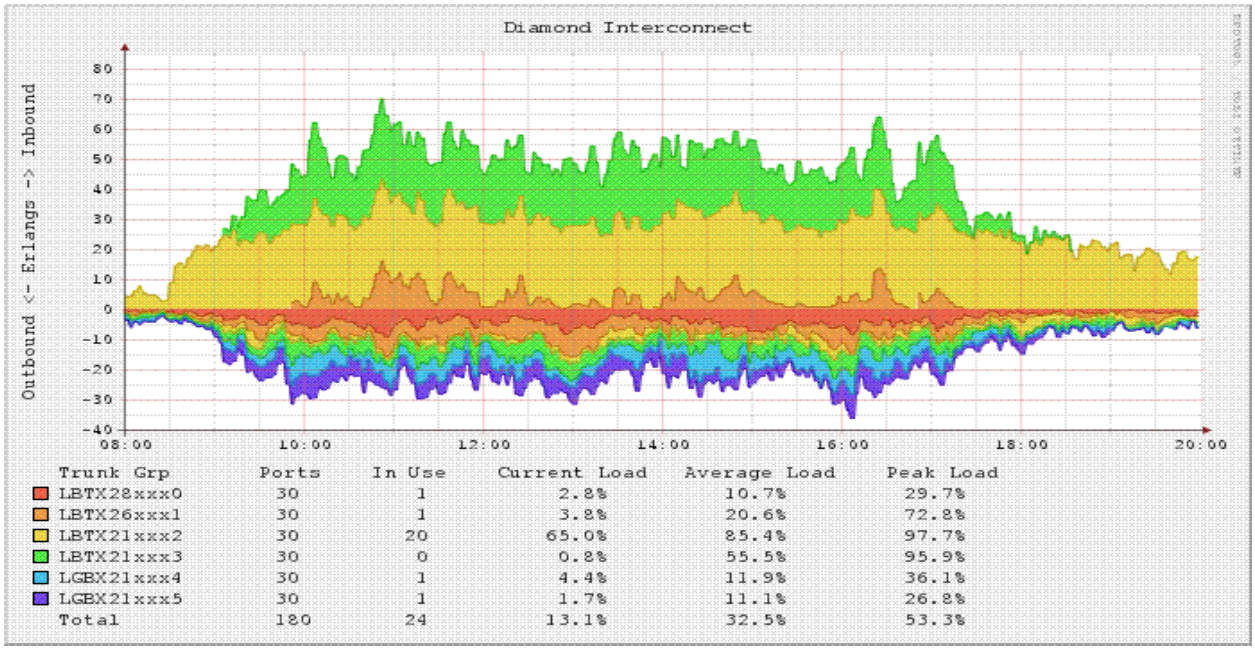
Welche Möglichkeiten und Vorteile bringt rrdtool

rrdtool ist ein Programm, mit dem zeitbezogene Messdaten gespeichert, zusammengefasst und visualisiert werden können. Das Programm wurde ursprünglich von Tobias Oetiker entwickelt und unter der GNU General Public License (GPL) lizenziert. Durch die Veröffentlichung als Freie Software haben inzwischen viele weitere Autoren neue Funktionalität und Fehlerbehebungen beigesteuert. rrdtool ist als Quelltext und als ausführbares Programm für viele Betriebssysteme verfügbar.

Quelle: http://de.wikipedia.org/wiki/RRDtool

Beispielgrafiken:





Quelle - http://www.mrtg.org/

Was wird benötigt: Hardware,Software, Tools etc.

SPS mit OSCAT Network-lib

Einen möglichst stromsparenden PC wegen dem Dauerbetrieb (24/7).

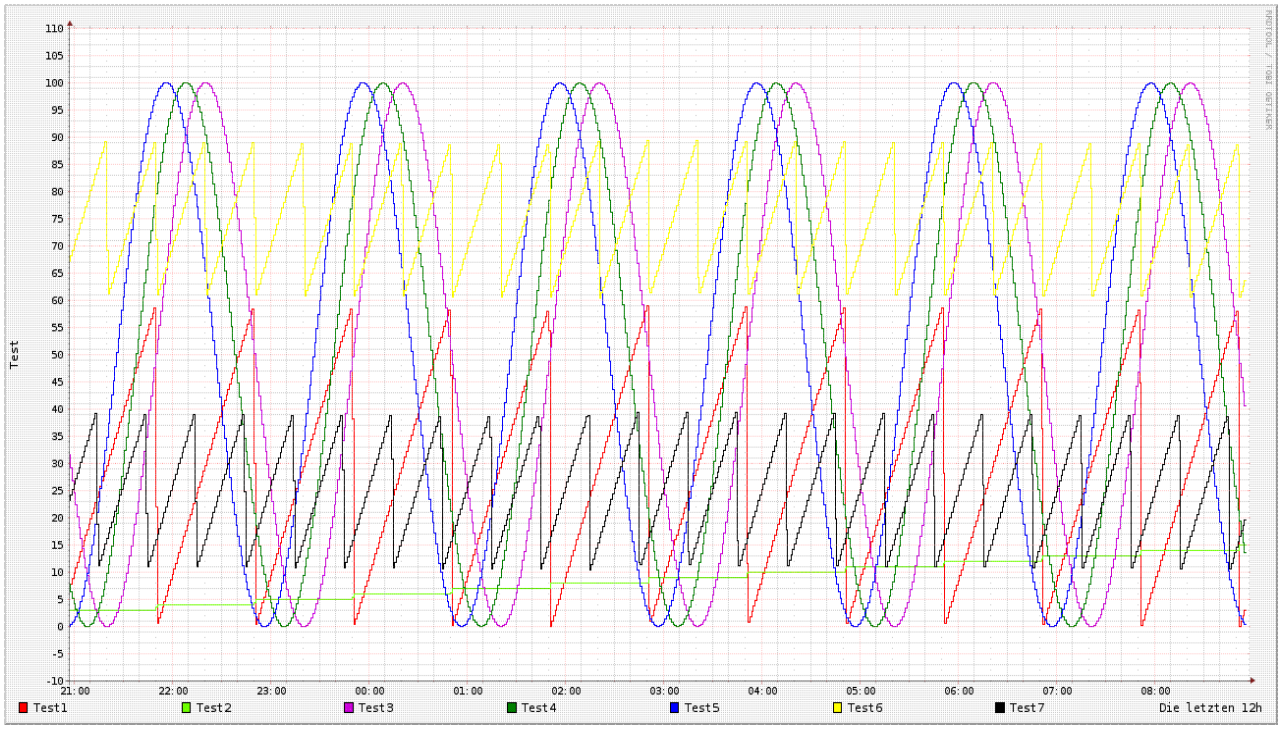
Auf dem PC müssen rrdtol und die php-Scripts installiert werden.

Die Scripts wurden auf einem Linux - XUbuntu - PC mit PHP entwickelt.

Quickstart:

- Ein Beispielprogramm mit dem einige Werte aufgezeichnet werden können, ist in der Oscat-network.lib unter demo/DLOG_RRD_DEMO zu finden.
- rrdtool Installation auf einem XUbuntu (Debian) PC entweder mit der Synaptic Paketverwaltung rrdtool auswählen und installieren, oder in der Konsole mit "apt-get install rrdtool" wird rrdtools installiert
- Script: create_test_rrd_db.php = erzeugt eine neue rrd-Datenbank, und muss bei Bedarf einmalig angepasst werden.
- Script: test_rrd.php = dieses Script wird von der SPS mit dem OSCAT Funktionsbaustein per HTTP-GET aufgerufen. Muss im Normalfall nicht angepasst werden, und gibt einen Fehlercode aus. Wenn Fehlerfrei dann wird eine "0" ausgegeben.
- Script: chart_test.php = Script um sich den Charts aus der rrd-DB zu erstellen und auf einer Webseite anzuzeigen. Muss im Normalfall nicht angepasst werden und gibt einen Fehlercode aus. Wenn Fehlerfrei dann wird eine "0" ausgegeben.
- Die drei php-Scripte in den gewünschten Ordner auf dem PC übertragen z.B. /var/www/rrd/ und auf die passenden Zugriffsrechte nicht vergessen.

Das Demo-Programm in Verbindung mit den Demo-php-Scripte erzeugen folgende Daten bzw. Grafik.



Links

http://www.mrtg.org/rrdtool/

http://de.wikipedia.org/wiki/RRDtool

http://www.rrze.uni-erlangen.de/dienste/arbeiten-rechnen/linux/howtos/rrdtool.shtml

http://arbeitsplatzvernichtung-durch-outsourcing.de/marty44/rrdtool.html

php-script – Beispiele / Vorlagen

create_test_rrddb.php

```
#!/etc/php5/cli -q
<?php
error_reporting(E_ALL);
# ==================================
# Erstellt eine rrd-Datenbank
# wird einmal von der Console aufgerufen
# 12.11.2010 by NetFritz
# ==================================
# create wp.rrd legt die Datenbank test.rrd an
# --step 60 alle 60sec wird ein wert erwartet
# DS:t1:GAUGE:120:0:100 es wird eine Datenquelle mit dem Namen t1 angelegt,
#    der Typ ist Gauge est wird 120sec gewartet auf neue Daten wenn nicht
#    werden die Daten als UNKNOWN in die Datenbank geschrieben
#    der minimale und Maximale Messwert
# RRA:AVERAGE:0.5:1:2160 Das ist rrd-Archiv  AVERAGE=Mittelwert 0.5= Inter-
vallabweichung
#     36h Archiv jede Minute ein Wert, 1:2160 = 1h=3600sec 36h*3600=129600
1Minute=60sec jede Minute ein Wert, 129600/60=2160 EintrÃ¤ge
#    RRA:AVERAGE:0.5:5:2016    1Woche    Archiv    alle    5Minuten    1Wert,
3600*24h*7Tage=604800Sec / (5Minuten+60Sec=2016 Einträge
# RRA:AVERAGE:0.5:15:2880 30Tage Archiv alle 15Minuten 1Wert,
# RRA:AVERAGE:0.5:60:8760 1Jahr Archiv alle 60Minuten ein Wert
# jetzt geht es los
$command="rrdtool create test.rrd \
         --step 60 \
         DS:t1:GAUGE:120:0:100 \
         DS:t2:GAUGE:120:0:100 \
         DS:t3:GAUGE:120:0:100 \
         DS:t4:GAUGE:120:0:100 \
         DS:t5:GAUGE:120:0:100 \
         DS:t6:GAUGE:120:0:100 \
         DS:t7:GAUGE:120:0:100 \
         RRA:AVERAGE:0.5:1:2160 \
         RRA:AVERAGE:0.5:5:2016 \
         RRA:AVERAGE:0.5:15:2880 \
         RRA:AVERAGE:0.5:60:8760";
```

```
system($command);
?>
```

test_rrd.php

```
<?php
# wird von Steuerung so aufgerufen
# http://mein_server/test_rrd.php?rrd_db=test.rrd&value=10:20:30:40:50:60
   $rrd_db = urldecode($_GET['rrd_db']);  # Name der RRD Datenbank
   $value = urldecode($_GET['value']);    # übergebene Werte
   # $array_value = explode(":",$value);
   # echo "$rrd_db <br>";
   # print_r($array_value);
   # echo "<br>";
   $commando = "/usr/bin/rrdtool update " . $rrd_db . " N:" . $value;
   system($commando,$fehler);
   echo $fehler . $commando;
?>
```

chart_test_rrd.php

```
<?php
// erstellt Chart fuer Test Werte, und wird vom Browser aufgerufen
$command="/usr/bin/rrdtool graph test0.png \
          --vertical-label=Test \
          --start end-12h \
          --width 600 \
          --height 200 \
          --alt-autoscale \
DEF:t1=test.rrd:t1:AVERAGE \
DEF:t2=test.rrd:t2:AVERAGE \
DEF:t3=test.rrd:t3:AVERAGE \
DEF:t4=test.rrd:t4:AVERAGE \
DEF:t5=test.rrd:t5:AVERAGE \
DEF:t6=test.rrd:t6:AVERAGE \
DEF:t7=test.rrd:t7:AVERAGE \
LINE1:t1#FF0000:Test1 \
LINE1:t2#6EFF00:Test2 \
LINE1:t3#CD04DB:Test3 \
```

```
LINE1:t4#008000:Test4 \
LINE1:t5#0000FF:Test5 \
LINE1:t6#0000FF:Test6 \
LINE1:t7#0000FF:Test7 \
          COMMENT:'Die letzten 12h'";
system($command);

echo "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
          \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n";
echo "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
echo "  <head>\n";
echo "    <title>Test</title>\n";
echo "  </head>\n";
echo "  <body>\n";
echo ("<center><img src='test0.png'></center>\n");
echo "    <center>Die letzten 12h</center>\n";
# echo "Fehler=" . $fehler;
echo "  </body>\n";
echo "</html>\n";
?>
```

7.11. DLOG_FILE_TO_FTP

Type	Funktionsbaustein:
IN_OUT	X : DLOG_DATA (DLOG Datenstruktur)
INPUT	FTP_URL : STRING(STRING_LENGTH) (FTP Zugriffspfad)
	FTP_ACTIV : BOOL (PASSIV = 0 / ACTIV = 1)
	FILE_DELETE : BOOL (Datei nach Übertragung löschen)
	TIMEOUT : TIME (Zeitüberwachung)
	RETRY : INT (Anzahl der Wiederholungen)
	RETRY_TIME : TIME (Wartezeit vor Wiederholung)
	DNS_IP4 : DWORD (IP4-Adresse des DNS-Server)

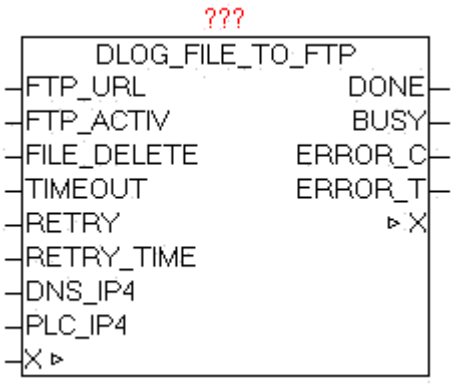
PLC_IP4 : DWORD (IP4-Adresse der eigenen Steuerung)

OUTPUT DONE : BOOL (Transfer ohne Fehler beendet)

BUSY : BOOL (Transfer ist aktiv)

ERROR_C : DWORD (Fehlercode)

ERROR_T : BYTE (Fehlertype)



Der Baustein DLOG_FILE_TO_FTP dient zum automatischen Übertragen der vom DLOG_STORE_FILE_CSV erzeugt Dateien auf einen FTP-Server.Der Parameter FTP_URL enthält den Namen des FTP-Server und optional den Benutzernamen und das Passwort, einen Zugriffspfad und eine zusätzliche Port-Nummer für den Datenkanal.Wird kein Benutzername bzw. Passwort übergeben, so versucht der Baustein sich automatisch als „Anonymous“ anzumelden. Der Parameter FTP_ACTIV bestimmt ob der FTP-Server im AKTIV oder PASSIV Modus betrieben wird. Im ACTIV Modus versucht der FTP-Server den Datenkanal zur Steuerung aufzubauen, dabei kann es jedoch durch Sicherheitssoftware, Firewall etc. zu Problemen kommen , da diese den Verbindungswunsch blockieren könnte. Hierzu müsste in der Firewall eine dementsprechende Ausnahmeregel definiert werden. Beim PASSIV Modus wird dieses Problem entschärft, da die Steuerung die Verbindung aufbaut, und somit problemlos die Firewall passieren kann. Der Steuerkanal wird immer über PORT 20 aufgebaut, und der Datenkanal standardmäßig über PORT21, dies ist aber wiederum abhängig ob ACTIV oder PASSIV Mode genutzt wird, bzw. optional eine PORT-Nummer in der FTP-URL angegeben wurde. Mit dem Parameter FILE_DELETE kann bestimmt werden, ob die Quell-Datei nach erfolgreicher Übertragung gelöscht werden soll. Dies funktioniert auf FTP als auch auf der Steuerungsseite. Bei Vorgabe von FTP-Verzeichnissen ist das Verhalten FTP-Server abhängig, ob hierbei diese schon existieren müssen, oder automatisch angelegt werden. Im Normalfall sollten diese schon vorhanden sein. Die Größe der Dateien ist an sich unbegrenzt, jedoch gibt es hier praktische Limits: Speicher auf SPS, FTP-Speicher und der Faktor Übertragungszeit. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn in der FTP-URL ein DNS-Name angegeben wird, alternativ kann in der FTP-URL auch eine IP-Adresse eingetragen werden.

Bei Parameter PLC_IP4 muss die eigene IP-Adresse angegeben werden. Sollten bei der Übertragung Fehler auftreten werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY = TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE = TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

Wenn Parameter RETRY = 0 ist , dann wird der FTP-Transfer solange wiederholt bis dieser erfolgreich abgeschlossen wurde. Wird bei RETRY ein Wert > 0 angeben , so wird der FTP-Transfer genauso oft bei Übertragungsstörung wiederholt. Danach wird dieser Auftrag einfach verworfen, und mit der nächsten Datei wird fortgefahren. Mittels RETRY-TIME kann die Wartezeit zwischen die Wiederholungen vorgegeben werden.

Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies standardmäßig notwendig wäre.

Grundlagen: http://de.wikipedia.org/wiki/File_Transfer_Protocol

URL-Beispiele:

ftp://benutzername:password@servername:portnummer/verzeichnis/

ftp://benutzername:password@servername

ftp://benutzername:password@servername/verzeichnis/

ftp://servername

ftp://benutzername:password@192.168.1.1/verzeichnis/

ftp://192.168.1.1

ERROR_T:

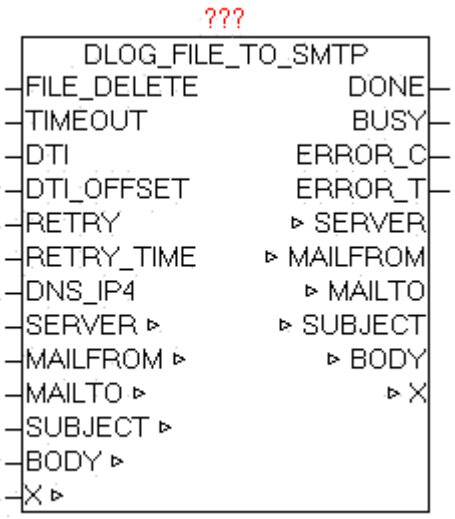
Wert	Eigenschaften
1	Störung: DNS_CLIENT Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Störung: FTP Steuerkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
3	Störung: FTP Datenkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUT ERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom FTP-Server empfangenen Response-Code. Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel: ERROR_T = 5 ERROR_C = 0x0028_00DC Ablauf-Schrittnummer 0x0028 = 40 Response-Code 0x00DC = 220

7.12. DLOG_FILE_TO_SMTP

Type Funktionsbaustein:

IN_OUT SERVER : STRING (URL des SMTP-Server)
MAILFROM : STRING (Absenderadresse)
MAILTO : STRING(STRING_LENGTH) (Empfängeradresse)
SUBJECT : STRING (Betreff-Text)
BODY : STRING(STRING_LENGTH) (Email-Inhalt)
FILES : STRING(STRING_LENGTH) (Dateien zum versenden)
X : DLOG_DATA (DLOG Datenstruktur)

INPUT	FILE_DELETE : BOOL (Datei nach Übertragung löschen)
	TIMEOUT : TIME (Zeitüberwachung)
	DTI : DT (aktuelles Datum-Uhrzeit)
	DTI_OFFSET : INT (Zeitzonen Offset zu UTC)
	RETRY : INT (Anzahl der Wiederholungen)
	RETRY_TIME : TIME (Wartezeit vor Wiederholung)
	DNS_IP4 : DWORD (IP4-Adresse des DNS-Server)
OUTPUT	DONE : BOOL (Transfer ohne Fehler beendet)
	BUSY : BOOL (Transfer ist aktiv)
	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)



Der Baustein DLOG_FILE_TO_SMTP dient zum automatischen Übertragen der vom DLOG_STORE_FILE_CSV erzeugt Dateien als Email an einen Email-Server.

Der Baustein verwendet intern den SMTP_CLIENT zum versenden

Der Parameter SERVER enthält den Namen des SMTP-Server und optional den Benutzernamen und das Passwort und eine Port-Nummer. Wird kein Benutzername bzw. Passwort übergeben, so wird nach Standard SMTP vorgegangen.

SERVER: URL-Beispiele:

benutzername:password@smtp_server

benutzername:password@smtp_server:portnummer

smtp_server

Sonderfall:

Befindet sich im Benutzernamen ein '@' so muss dies als '%' Zeichen übergeben werden, und wird danach vom Baustein automatisch wieder korrigiert.

Bei Angabe von Benutzer und Passwort wird Extend-SMTP benutzt, und automatisch das möglichst sicherste Authentifizierungs-Verfahren angewendet. Bei Parameter MAILFROM wird die Absenderadresse angeben:

z.B. oscat@gmx.net

optional kann ein zusätzlicher „Angezeigter Name“ hinzugefügt werden. Dieser wird vom Email-Client automatisch anstatt der echten Absenderadresse angezeigt. Damit kann immer ein leicht erkennbarer Name angewendet werden.

z.B. oscat@gmx.net;Station_01

Der Email-Client zeigt als Absender dann „Station_01“ an. Somit können mehrere Teilnehmer die gleiche Email-Adresse benutzen, jedoch eine eigenen „Alias“ Kennung mitsenden.

Bei Parameter MAILTO können To,Cc,Bc angeben werden. Die verschiedene Empfängergruppen werden mittels '#' als Trennzeichen als Liste angegeben. Mehrere Adressen innerhalb der selben Gruppe werden mit dem Trennzeichen ';' unterteilt. Es können von jeder Gruppe beliebig viele Empfänger vorgegeben werden, einzige Beschränkung ist die Länge des MAILTO-Strings.

To;To..#Cc;Cc...#Bc;Bc...

Beispiele.

o1@gmx.net;o2@gmx.net#o1@gmx.net#o2@gmx.net

definiert zwei To-Adressen, eine Cc-Adresse und eine Bc-Adresse

##o2@gmx.net

definiert nur eine Bc-Adresse

Mit SUBJECT kann ein Betreff-Text vorgeben werden, sowie bei BODY ein Email Inhalt als Text. Der aktuelle Date/Time Wert muss bei DTI , und bei DTI_OFFSET der Korrekturwert als Offset in Minuten zur UTC (Weltzeit) angegeben werden. Wenn bei DTI die Weltzeit UTC übergeben wird, muss bei DTI_OFFSET 0 übergeben werden.

Die Überwachungszeit kann bei Parameter TIMEOUT vorgegeben werden. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn bei SERVER ein DNS-Name angegeben wird.Sollten bei der Übertragung Fehler auftreten, werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY = TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE = TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

Wenn Parameter RETRY = 0 ist , dann wird der SMTP-Transfer solange wiederholt bis dieser erfolgreich abgeschlossen wurde. Wird bei RETRY ein Wert > 0 angeben , so wird der SMTP-Transfer genauso oft bei Übertragungsstörung wiederholt. Danach wird dieser Auftrag einfach verworfen, und mit der nächsten Datei wird fortgefahren. Mittels RETRY-TIME kann die Wartezeit zwischen die Wiederholungen vorgegeben werden.

Mit dem Parameter FILE_DELETE = TRUE wird die Datei nach erfolgreicher Übertragung per Email, auf der Steuerung gelöscht.

Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies standardmäßig bei der notwendig wäre.

Grundlagen:

http://de.wikipedia.org/wiki/SMTP-Auth

http://de.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

ERROR_T:

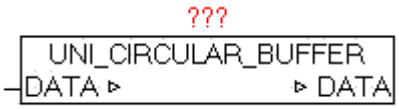
Wert	Eigenschaften
1	Störung: DNS_CLIENT Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Störung: SMTP Steuerkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUT ERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom SMTP-Server empfangenen Response-Code. Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel: ERROR_T = 5

	ERROR_C = 0x0028_00FA Ablauf-Schrittnummer 0x0028 = 40 Response-Code 0x00DC = 250

7.13. UNI_CIRCULAR_BUFFER

Type Funktionsbaustein:

IN_OUT DATA : UNI_CIRCULAR_BUFFER_DATA (Datenspeicher)



Der Baustein UNI_CIRCULAR_BUFFER ist ein RING Buffer der im FIFO (First in – First out) Prinzip arbeitet, und beliebige Daten als Bytestream verarbeiten kann.

Hierzu kann über die Datenstruktur UNI_CIRCULAR_BUFFER_DATA alles abgewickelt werden.

Folgende Kommandos über DATA.D_MODE werden unterstützt.

01	Element in Buffer schreiben
10	Element von Buffer lesen, aber nicht entfernen
11	Das zuvor mit Kommando 10 gelesen Element wird entfernt.
12	Element von Buffer lesen, und entfernen
99	Buffer wird rückgesetzt. Alle Daten werden gelöscht

Mittels DATA.D_HEAD (WORD) kann im rechten Byte der Elemente-Typ, und im linken Byte optional eine zusätzliche Anwender-Kennung vorgeben werden.

D_HEAD = LEFT-BYTE (ADD-Ino) , RIGHT-BYTE (Type-Kennung)

Type-Kennungen:

1 = STRING (Bei DATA.D_STRING muss der STRING übergeben werden)

2 = REAL (Bei DATA.D_REAL muss der REAL-Wert übergeben werden)

3 = DWORD (Bei DATA.D_DWORD muss das DWORD übergeben werden)

X = Header-Information ohne Daten

Bei DATA.BUF_SIZE wird die Anzahl der Bytes ausgegeben die die abgelegten Elemente in Summe belegen. Mit DATA.BUF_COUNT wird die Anzahl der sich im Buffer befindlichen Elemente ausgegeben. Sowie über BUF_USED wird die Belegung des Buffer als Prozent-Wert ausgeben.

Wird ein Element in den Buffer geschrieben , und der notwendige freie Platz (Speicher) ist nicht vorhanden, bleibt nach Aufruf des Baustein DATA.D_MODE unverändert. Nur wenn D_MODE nach Bausteinaufruf 0 enthält, wurde der Befehl erfolgreich durchgeführt.

Beim Lesen von Elemente gilt die gleiche Funktionsweise.

Nur wenn D_MODE danach 0 ist, kann über D_HEAD die Datentype festgestellt werden, und wenn notwendig, die Daten von D_STRING,D_REAL,D_DWORD entnommen werden. Nach erfolgreichen Lesen muss noch das Löschen des Elementes mit Befehl 11 durchgeführt werden.

Beispiel: Element schreiben:

```
DATA.D_MODE := 1; (* Befehl Daten schreiben *)
DATA.D_HEAD :=  1; (* Element-Type = STRING *)
DATA.D_STRING := 'Das ist der Text';
Baustein-Aufruf()
```

wenn danach DATA.D_MODE = 0 ist , wurde das Element erfolgreich gespeichert

Beispiel: Element lesen:

```
DATA.D_MODE := 10; (* Befehl Element lesen *)
Baustein-Aufruf()
```

Ergebnis

```
DATA.D_HEAD =  1; (* Element-Type = STRING *)
DATA.D_STRING = 'Das ist der Text';
DATA.D_MODE = 0
```

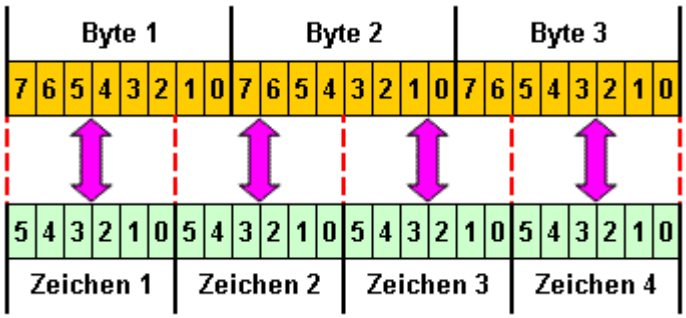
Beispiel: Element löschen:

```
DATA.D_MODE := 11; (* Befehl Element löschen *)
Baustein-Aufruf()
DATA.D_MODE = 0; (* Element wurde gelöscht *)
```

8. Konverter

8.1. BASE64

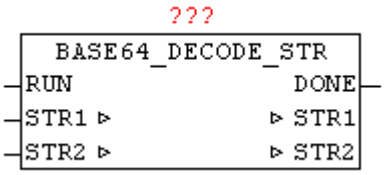
Die BASE64 Kodierung ist ein Verfahren zur Kodierung von 8-Bit-Binärdaten , in eine Zeichenfolge, die nur aus 64 weltweit verfügbaren ASCII-Zeichen besteht. Einsatzgebiet ist HTTP "Basic" Authentication, PGP Signaturen und Schlüssel und MIME Kodierung bei E-Mail. Um z.B. beim SMTP Protokoll den problemlosen Transport der Binärdaten zu ermöglichen, ist eine Konvertierung notwendig, da in der Originalfassung nur 7-Bit-ASCII-Zeichen vorgesehen waren.



Bei der Kodierung werden immer drei Byte des Bytestroms (24 bit) in vier 6-bit-Blöcke zerlegt. Jeder dieser 6-bit-Blöcke ergibt eine Zahl zwischen 0 und 63. Dies ergibt folgende 64 druckbare ASCII-Zeichen [A–Z], [a–z], [0–9], [+/]. Bei der Kodierung erhöht sich der Platzbedarf des Datenstroms um 33%, also aus je 3 Zeichen werden je 4 Zeichen. Ergibt die Länge der Kodierung kein durch 4 teilbares Ergebnis werden am Ende Füllzeichen angehängt. Dabei wird das Zeichen „=“ verwendet

8.2. BASE64_DECODE_STR

Type	Funktionsbaustein
Input	RUN : BOOL (Positive Flanke startet die Konvertierung)
Output	DONE : BOOL (TRUE wenn Konvertierung beendet ist)
I/O	STR1 : STRING(192) (Text im Format BASE64)
	STR2: STRING(144) (konvertierter normaler Text)



Mit BASE64_DECODE_STR kann ein in BASE64 kodierten Text wieder in normaler Text konvertiert werden. Mit positiver Flanke von RUN wird der Vorgang gestartet. Dabei wird DONE sofort rückgesetzt, sollte es von einer vorhergehenden Konvertierung noch gesetzt gewesen sein. Der BASE64 kodierte Text wird über STR1 übergeben, und nach Beendigung der Konvertierung steht der normaler Text bei STR2 zur Verfügung, und DONE wird auf TRUE gesetzt.

Beispiel:

Text in STR1

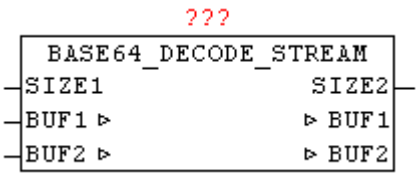
'T3BlbiBTb3VyY2UgQ29tbXVuaXR5IGZvciBBdXRvbWF0aW9uIFRlY2hub2xvZ3k='

Ergebnis in STR2

Text in STR2 = 'Open Source Community for Automation Technology'

8.3. BASE64_DECODE_STREAM

Type	Funktionsbaustein
Input	SIZE1 : INT (Anzahl Bytes im BUF1 zum dekodieren)
Output	SIZE2 : INT (Anzahl Bytes im BUF2 des dekodierten Ergebnisses)
I/O	BUF1 : ARRAY[0..63] OF BYTE (BASE64 Daten zum konvertieren)
	BUF2 : ARRAY[0..47] OF BYTE (konvertierte Daten)

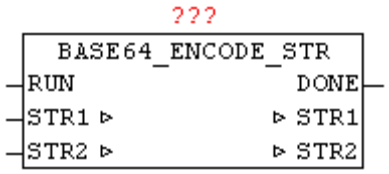


Mit BASE64_DECODE_STREAM kann ein beliebig langer BASE64 Byte-

Datenstrom dekodiert werden. In einem Durchlauf können bis zu 64 Byte dekodiert werden, die wiederum als Ergebnis je maximal 48 Bytes ergeben. Dabei werden die Quelldaten dem Decoder über BUF1 im Datenstrom verfahren als einzelne Datenblöcke zugeführt, und in dekodierter Form von diesem wieder bei BUF2 ausgegeben. Für die Weiterverarbeitung der BUF2 Daten bevor der nächste Datenblock konvertiert wird, hat der Anwender zu sorgen. Die Anzahl der Bytes im BUF2 wird mittels SIZE2 vom Baustein ausgegeben.

8.4. BASE64_ENCODE_STR

ype	Funktionsbaustein
Input	RUN : BOOL (Positive Flanke startet die Konvertierung)
Output	DONE : BOOL (TRUE wenn Konvertierung beendet ist)
I/O	STR1 : STRING(144) (Text zum konvertieren)
	STR2 : STRING(192) (konvertierter Text in Format BASE64)



Mit BASE64_ENCODE_STR kann ein Standard-Text in einen BASE64 kodierten Text konvertiert werden. Mit positiver Flanke von RUN wird der Vorgang gestartet. Dabei wird DONE sofort rückgesetzt, sollte es von einer vorhergehenden Konvertierung noch gesetzt gewesen sein. Der zu konvertierende Text wird über STR1 übergeben, und nach Beendigung der Konvertierung steht die BASE64 Kodierung bei STR2 zur Verfügung, und DONE wird auf TRUE gesetzt.

Beispiel:

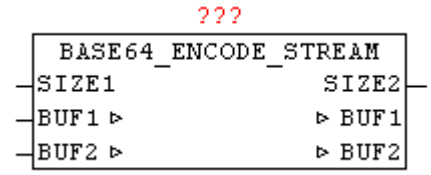
Text in STR1 = 'Open Source Community for Automation Technology'

Ergebnis in STR2

'T3BlbiBTb3VyY2UgQ29tbXVuaXR5IGZvciBBdXRvbWF0aW9uIFRlY2hub2xvZ3k='

8.5. BASE64_ENCODE_STREAM

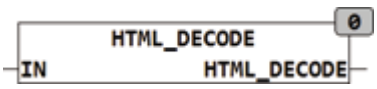
Type	Funktionsbaustein
Input	SIZE1 : INT (Anzahl Bytes im BUF1 zum kodieren)
Output	SIZE2 : INT (Anzahl Bytes im BUF2 des kodierten Ergebnisses)
I/O	BUF1 : ARRAY[0..47] OF BYTE (Daten zum konvertieren)
	BUF2 : ARRAY[0..63] OF BYTE (BASE64 konvertierte Daten)



Mit BASE64_ENCODE_STREAM kann ein beliebig langer Byte-Datenstrom nach BASE64 kodiert werden. In einem Durchlauf können bis zu 48 Byte konvertiert werden, die wiederum maximal 64 Bytes ergeben. Dabei werden die Quelldaten dem Encoder über BUF1 im Datenstrom verfahren als einzelne Datenblöcke zugeführt, und in kodierter Form von diesem wieder bei BUF2 ausgegeben. Für die Weiterverarbeitung der BUF2 Daten bevor der nächste Datenblock konvertiert wird, hat der Anwender zu sorgen. Die Anzahl der Bytes im BUF2 wird mittels SIZE2 vom Baustein ausgegeben.

8.6. HTML_DECODE

Type	Funktion : STRING(STRING_LENGTH)
Input	IN : STRING (Zeichenkette)
Output	STRING(STRING_LENGTH) (Zeichenkette)



HTML_DECODE wandelt reservierte Zeichen welche in der Form &name; im HTML Code gespeichert sind in die originalen Zeichen um. Zusätzlich werden alle codierten Sonderzeichen in den entsprechenden ASCII Code umgewandelt. Sonderzeichen können in HTML durch folgende Zeichenfolge repräsentiert sein:

- &#NN; wobei NN die Position des Zeichens innerhalb der Zeichentabelle in dezimaler Schreibweise darstellt.

- &#xNN; oder &#XNN wobei NN die Position des Zeichens innerhalb der Zeichentabelle in hexadezimaler Schreibweise darstellt.

&Name; Sonderzeichen haben Namen wie zum Beispiel € für €.

Die reservierten Zeichen in HTML sind:

& wird codiert als &

> wird codiert als >

< wird codiert als <

" wird codiert als "

Beispiele:

HTML_DECODE('1 ist >als 0') = '1 ist > als 0';

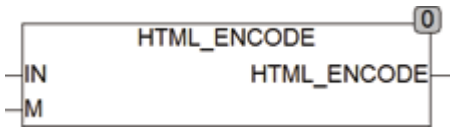
HTML_DECODE('&#D79;&#D83;&#D67;&#D65;&#D84;') = 'OSCAT';

HTML_DECODE('&#xH4F;&#xH53;&#xH43;&#xH41;&#xH54;') = 'OSCAT';

HTML_DECODE('&#XH4F;&#XH53;&#XH43;&#XH41;&#XH54;') = 'OSCAT';

8.7. HTML_ENCODE

Type	Funktion : STRING(STRING_LENGTH)
Input	IN : STRING (Zeichenkette)
	M : BOOL (Mode)
Output	STRING(STRING_LENGTH) (Zeichenkette)



HTML_ENCODE wandelt in HTML reservierte Zeichen um in die Form &Name;. Wird der Eingang M auf TRUE gesetzt werden zusätzlich alle Zeichen mit dem Code 160-255 und 128 in die &Name Konvention umgesetzt.

Vorsicht ist bei der Anwendung von Zeichensätzen geboten weil diese nicht auf allen Systemen identisch sind und Abweichungen speziell bei Sonderzeichen häufig vorkommen. So ist zum Beispiel nicht bei allen Systemen das € Zeichen auf Position 128 in der Zeichentabelle.

Die reservierten Zeichen in HTML sind:

& wird codiert als &

> wird codiert als >

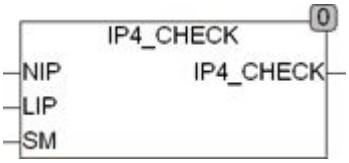
< wird codiert als <

" wird codiert als "

HTML_ENCODE wandelt die Zeichenkette '1 ist > als 0' um in '1 ist >als 0'.

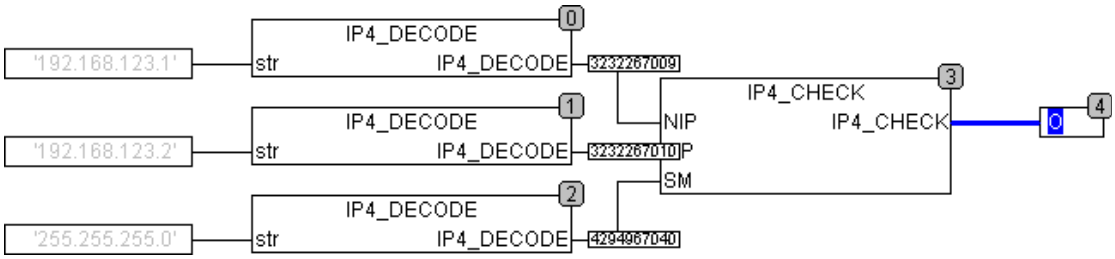
8.8. IP4_CHECK

Type	Funktion : BOOL
Input	NIP : DWORD (Netzwerk IP Adresse)
	LIP : DWORD (Lokale IP Adresse)
	SM : DWORD (Subnet Maske)
Output	BOOL (TRUE wenn NIP und LIP im gleichen Subnet liegen)



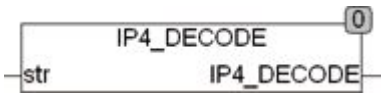
IP4_CHECK prüft ob eine Netzwerkadresse NIP und die Lokale Adresse LIP im gleichen Subnet liegen. Beiden Adressen werden zuerst mit der Subnet Maske Maskiert und dann auf Gleichheit geprüft. Es werden nur die Bits auf Gleichheit geprüft die in der Subnet Maske TRUE sind. Die Netzwerk Adressen müssen dem Ipv4 Format entsprechen und als DWORD vorliegen. Sollen IP Adressen die als String vorliegen geprüft werden müssen diese vorher in DWORD gewandelt werden.

Folgendes Beispiel zeigt 2 IP Adressen und eine Subnet Maske die als String vorliegen und nach entsprechender Umwandlung in DWORD geprüft werden. Der Ausgang wird TRUE weil beide Adressen im gleichen Subnet liegen.



8.9. IP4_DECODE

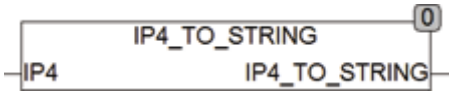
Type	Funktion : DWORD
Input	STR : STRING(15) (Zeichenkette die die IP Adresse enthält)
Output	DWORD (dekodierte IP v4 Adresse)



IP4_DECODE dekodiert die in STR enthaltene Zeichenkette als IP v4 Adresse und gibt diese als DWORD zurück. Eine Rückgabe von 0 bedeutet das eine ungültige Adresse oder die Adresse '0.0.0.0' ausgewertet wurde. IP4 kann auch zum Auswerten einer Subnet Maske des IP v4 Formates verwendet werden.

8.10. IP4_TO_STRING

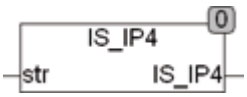
Type	Funktion : STRING(15)
Input	IP4 : BOOL (Zeichenkette die die IP Adresse enthält)
Output	DWORD (dekodierte IP v4 Adresse)



IP4_TO_STRING wandelt die als DWORD gespeicherte IP4 Adresse in IN in einen STRING um. Das Format entspricht 'NNN.NNN.NNN.NNN'.

8.11. IS_IP4

Type	Funktion : BOOL
Input	STR : STRING (zu prüfende Zeichenkette)
Output	BOOL (TRUE wenn STR eine gültige IP v4 Adresse enthält)



IS_IP4 prüft ob die Zeichenkette STR eine gültige IP v4 Adresse enthält, wenn nicht wird FALSE zurückgegeben. Ein gültige IP v4 Adresse besteht

aus 4 Zahlen von 0 – 255 die mit je einem Punkt getrennt sind. Die Adresse 0.0.0.0 wird dabei als falsch eingestuft.

IS_IP4(0.0.0.0) = FALSE

IS_IP4(255.255.255.255) = TRUE

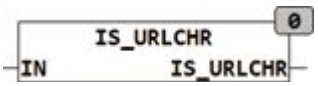
IS_IP4(256.255.255.255) = FALSE

IS_IP4(0.1.2.) = FALSE

IS_IP4(0.1.2.3.) = FALSE

8.12. IS_URLCHR

Type	Funktion : BOOL
Input	IN : STRING (zu prüfende Zeichenkette)
Output	BOOL (TRUE wenn STR eine gültige IP v4 Adresse enthält)



IS_URLCHR prüft ob die Zeichenkette nur zulässige Zeichen für eine URL Kodierung enthält. Enthält die Zeichenkette reservierte Zeichen gibt Sie FALSE zurück, andernfalls TRUE.

Für eine URL sind folgende Zeichen zulässig:

[A..Z]
[a..z]
[0..9]
[-._~]

alle anderen Zeichen sind reserviert oder nicht zulässig.

8.13. MD5_AUX

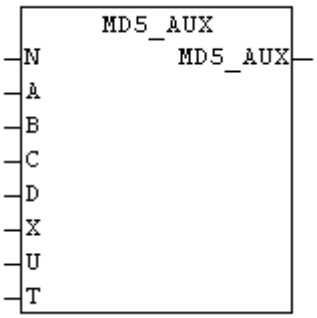
Type	Funktion : DWORD
Input	N : INT (Interne Verwendung)
	A : DWORD (Interne Verwendung)
	B : DWORD (Interne Verwendung)
	C : DWORD (Interne Verwendung)

D : DWORD (Interne Verwendung)

X : DWORD (Interne Verwendung)

U : INT (Interne Verwendung)

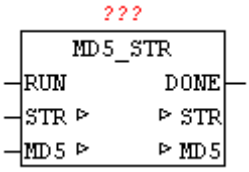
T : DWORD (Interne Verwendung)



Bei der MD5 Hash Erzeugung werden mehre sogenannte Runden durchlaufen, bei denen komplexe mathematische Berechnungen durchgeführt werden. Damit die Menge an redundanten Code im Baustein MD5_STREAM klein bleibt, sind wiederkehrende Berechnungen als Makro in den MD5_AUX verlagert worden . Diese Baustein hat nur in Verbindung mit dem Baustein MD5_STREAM eine sinnvolle Anwendung.

8.14. MD5_STR

Type	Funktionsbaustein
Input	RUN : BOOL (Positive Flanke startet die Berechnung)
Output	DONE : BOOL (TRUE wenn Berechnung beendet ist)
I/O	MD5 : ARRAY[0..15] OF BYTE (aktueller MD5 HASH)
	STR : STRING(STRING_LENGTH) (Text zur HASH-Bildung)



Mit MD5_STR kann von einen String der MD5 Hash berechnet werden. Bei

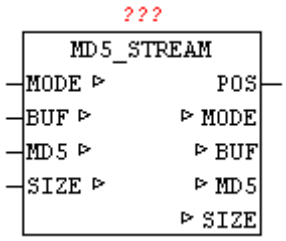
STR wird der „STR“ dem Baustein übergeben, und bei positiver Flanke am Eingang „RUN“ wird die Berechnung gestartet. Beim Start wird DONE sofort rückgesetzt, und nach Beendigung des Vorganges wird DONE auf TRUE gesetzt. Danach steht am Parameter HASH der aktuell berechnete HASH-Wert zur Verfügung. (Siehe Baustein MD5-STREAM).

Beispiel:

der MD5 Hash von 'Oscat' ist 30f33ddb9f17df7219e1acdea3386743

8.15. MD5_STREAM

Type	Funktionsbaustein
I/O	MODE : INT (Modus: 1=Init / 2=Datenblock / 3=Fertig)
	BUF : ARRAY[0..63] OF BYTES (Quelldaten)
	MD5 : ARRAY[0..15] OF BYTE (aktueller MD5-HASH)
	SIZE : UDINT (Anzahl der Daten)
Output	POS : UDINT (Startadresse des angeforderten Datenblocks)



Der Baustein MD5_STREAM ermöglicht die Berechnung des MD5 (*Message-Digest Algorithm 5*) einer kryptographischen Hash-Funktion.

Damit lässt sich von einem beliebigen Datenstrom ein eindeutiger Prüfwert erstellen. Es ist praktisch unmöglich, zwei verschiedene Nachrichten mit dem gleichen Prüfwert zu finden, dies wird auch als Kollisionsfreiheit bezeichnet. Damit lässt sich z.B. eine Konfigurationsdatei auf Veränderung oder Manipulation prüfen.

Mit dem Hash-Algorithmus (MD5) wird ein Hash-Wert von 128 Bit Länge für beliebige Daten erzeugt. Die maximale Länge des Datenstrom ist bei diesem Baustein auf 2^32 (4 Gigabyte) begrenzt. Das Ergebnis liegt bei Parameter MD5 als 16 Byte Hash-Wert vor.

Beispiel:

Es liegen 2000 Bytes in einen Buffer oder werden über das Filesystem Blockweise eingelesen

Anwender setzt MODE auf 1 und SIZE auf 2000. Aufruf des MD5_STREAM

MD5 STREAM führt eine Initialisierung durch und setzt MODE auf 2 und gibt bei POS den Index (Basis 0) der gewünschte Daten an. bei SIZE wird die Anzahl der Daten vorgegeben die in den Datenspeicher BUF kopiert werden sollen.

Anwender kopiert die angeforderten Daten in den BUF und ruft den Baustein MD5_STREAM wiederholt auf. dieser Schritt wird immer wiederholt solange MODE auf 2 bleibt.

Wenn der MD5_STREAM den letzten Datenblock verarbeitet hat, setzt dieser MODE auf 3. Es kann auch beim letzten Block vorkommen das bei SIZE die Länge null vorgegeben wird, somit müssen keine Daten in BUF kopiert werden.

Der aktuelle HASH-Wert kann nun als Ergebnis weiterverarbeitet werden.

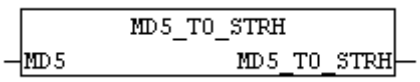
Beispiel:

der MD5 Hash von 'Oscat' ist 30f33ddb9f17df7219e1acdea3386743

8.16. MD5_TO_STRH

Type Funktion : STRING(32)

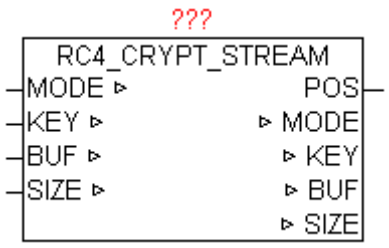
Input MD5 : ARRAY[0..15] OF BYTE (MD5-HASH)



Der Baustein MD5_TO_STRH konvertiert das MD5 Bytearray in einen Hex-String.

8.17. RC4_CRYPT_STREAM

Type	Funktionsbaustein
I/O	MODE : INT (Modus: 1=Init / 2=Datenblock / 3=Fertig)
	KEY : STRING(40) (320 Bit langer geheimer Schlüssel)
	BUF : ARRAY[0..63] OF BYTES (Datenblock zum verarbeiten)
	SIZE : UDINT (Anzahl der Daten)
Output	POS : UDINT (Startadresse des angeforderten Datenblocks)



Der Baustein RC4_CRYPT_STREAM nutzt die RC4 Datenstromchiffrierung um einen nahezu beliebig langen Datenstrom zu verarbeiten. Dieser Standard wird z.B. bei SSH 1, HTTPS und WEP bzw. WPA eingesetzt , und ist somit weit verbreitet. Der Algorithmus kann prinzipiell mit bis zu 2048 Bit langen Schlüssel arbeiten, jedoch ist dieser am Baustein auf einen 40 Zeichen langen Key begrenzt (kann aber jederzeit auf bis zu 250 Zeichen angepasst werden). Somit ergibt sich hier eine Schlüssellänge von 320 Bit , die für Anwendungen auf einer SPS mehr als ausreichend sind. Die maximale Länge des Datenstrom ist bei diesem Baustein auf 2^32 (4 Gigabyte) begrenzt. Der Baustein kann zum Verschlüsseln als auch zum Entschlüsseln von RC4 Daten benutzt werden. Pro Zyklus können immer 64 Bytes verarbeitet werden, diese werden im Blockmodus seriell verarbeitet. Die Daten die verschlüsselt bzw. entschlüsselt werden stehen nach Aufruf des Bausteins im BUF zur Weiterverarbeitung zur Verfügung, und müssen natürlich vor jedem neuen Datenblock vorher vom Anwender verarbeitet werden.

Beispiel:

Es liegen 2000 Bytes in einen Buffer oder werden über das Filesystem Blockweise eingelesen

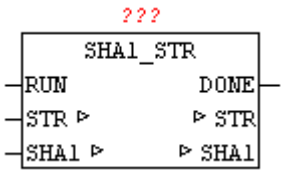
Anwender setzt MODE auf 1 und SIZE auf 2000. Aufruf des RC4_CRYPT_STREAM

RC4_CRYPT_STREAM führt eine Initialisierung durch und setzt MODE auf 2 und gibt bei POS den Index (Basis 0) der gewünschte Daten an. bei SIZE wird die Anzahl der Daten vorgegeben die in den Datenspeicher BUF kopiert werden sollen.

Anwender kopiert die angeforderten Daten in den BUF und ruft den Baustein RC4_CRYPT_STREAM wiederholt auf. dieser Schritt wird immer wiederholt solange MODE auf 2 bleibt. Wenn der RC4_CRYPT_STREAM den letzten Datenblock verarbeitet hat, setzt dieser MODE auf 3.

8.18. SHA1_STR

Type	Funktionsbaustein
Input	RUN : BOOL (Positive Flanke startet die Berechnung)
Output	DONE : BOOL (TRUE wenn Berechnung beendet ist)
	HASH : ARRAY[0..19] OF BYTE (aktueller SHA1-HASH)
I/O	STR : STRING(STRING_LENGTH) (Text zur HASH-Bildung)



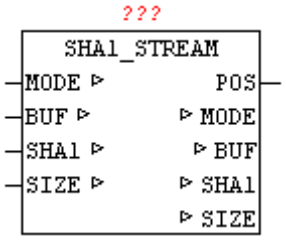
Mit SHA1_STR kann von einen String der SHA1 Hash berechnet werden. Bei STR wird der String dem Baustein übergeben, und bei positiver Flanke am Eingang „RUN“ wird die Berechnung gestartet. Beim Start wird DONE sofort rückgesetzt, und nach Beendigung des Vorganges wird DONE auf TRUE gesetzt. Danach steht am Parameter HASH der aktuell berechnete HASH-Wert zur Verfügung. (Siehe Baustein SHA1-STREAM).

Beispiel:

Hash-Wert von 'Oscat' ist 4fe4fa862f2e7b91bf95abe9f22831247a3afd35

8.19. SHA1_STREAM

Type	Funktionsbaustein
I/O	MODE : INT (Modus: 1=Init / 2=Datenblock / 3=Fertig)
	BUF : ARRAY[0..63] OF BYTES (Quelldaten)
	SHA1 : ARRAY[0..19] OF BYTE (aktueller SHA1-HASH)
	SIZE : UDINT (Anzahl der Daten)
Output	POS : UDINT (Startadresse des angeforderten Datenblocks)



Der Baustein SHA1_STREAM ermöglicht die Berechnung der standardisierten kryptologischen Hash-Funktion SHA-1 (Secure Hash Algorithm).

Damit lässt sich von einem beliebigen Datenstrom ein eindeutiger Prüfwert erstellen. Es ist praktisch unmöglich, zwei verschiedene Nachrichten mit dem gleichen Prüfwert zu finden, dies wird auch als Kollisionsfreiheit bezeichnet. Damit lässt sich z.B. eine Konfigurationsdatei auf Veränderung oder Manipulation prüfen.

Mit dem sicheren Hash-Algorithmus (SHA1) wird ein Hash-Wert von 160 Bit Länge für beliebige Daten erzeugt. Die maximale Länge des Datenstrom ist bei diesem Baustein auf 2^32 (4 Gigabyte) begrenzt. Das Ergebnis liegt als 20 Byte Hash-Wert vor, das als ARRAY [0..19] OF BYTE ausgegeben wird.

Beispiel:

Es liegen 2000 Bytes in einen Buffer oder werden über das Filesystem Blockweise eingelesen

Anwender setzt MODE auf 1 und SIZE auf 2000. Aufruf des SHA1_STREAM

SHA1_STREAM führt eine Initialisierung durch und setzt MODE auf 2 und gibt bei POS den Index (Basis 0) der gewünschte Daten an. bei SIZE wird die Anzahl der Daten vorgegeben die in den Datenspeicher BUF kopiert werden sollen.

Anwender kopiert die angeforderten Daten in den BUF und ruft den Baustein SHA1_STREAM wiederholt auf. dieser Schritt wird immer wiederholt solange MODE auf 2 bleibt.

Wenn der SHA1_STREAM den letzten Datenblock verarbeitet hat, setzt dieser MODE auf 3. Es kann auch beim letzten Block vorkommen das bei SIZE die Länge null vorgegeben wird, somit müssen keine Daten in BUF kopiert werden.

Der aktuelle HASH-Wert kann nun als Ergebnis weiterverarbeitet werden.

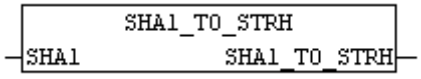
Beispiel:

Hash-Wert von 'Oscat' ist 4fe4fa862f2e7b91bf95abe9f22831247a3afd35

8.20. SHA1_TO_STRH

Type Funktion : STRING(40)

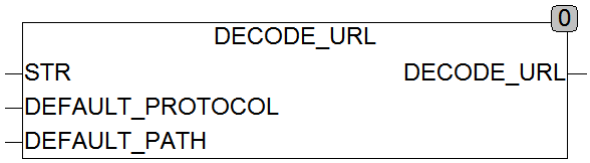
Input MD5 : ARRAY[0..19] OF BYTE (SHA1-HASH)



Der Baustein SHA1_TO_STRH konvertiert das SHA1 Bytearray in einen Hex-String.

8.21. STRING_TO_URL

Type	Funktion : URL
Input	STR : STRING(STRING_LENGTH) (Unified Resource Locator)
	DEFAULT_PROTOCOL : STRING (Ersatzprotokoll)
	DEFAULT_PATH : STRING (Ersatzpfad)
Output	URL (URL)



STRING_TO_URL zerlegt eine URL (Uniform Resource Locator) in seine Bestandteile und speichert diese in dem Datentyp URL. Wird in STR kein Pfad oder kein Protokoll spezifiziert, so setzt die Funktion die fehlenden Werte Automatisch mit den spezifizierten Ersatzwerten.

Eine URL setzt sich wie folgt zusammen:

protokoll://user:passwort@domain:port/path?query#anker

Beispiel: ftp://hugo:nono@oscat.de:1234/download/manual.html

einige Bestandteile der URL sind optional wie zum Beispiel User, Passwort, Anker, Query ...

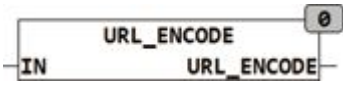
8.22. URL_DECODE

Type	Funktion : STRING(STRING_LENGTH)
Input	IN : STRING (Zeichenkette)
Output	STRING(STRING_LENGTH) (Zeichenkette)

URL_DECODE wandelt die als %HH codierten Sonderzeichen in der Zeichenkette IN in den entsprechenden ASCII Code um. In einer URL Kodierung dürfen nur die Zeichen [A..Z, a..z, 0..9, -._~] vorkommen. Anderen Zeichen können mit einem % Zeichen gefolgt vom 2 Zeichen langen Hexadezimalcode des Zeichens dargestellt werden. Das reservierte Zeichen '#' wird dabei als '%23' kodiert.

8.23. URL_ENCODE

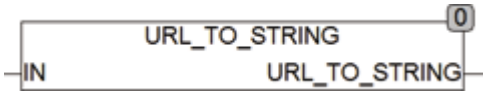
Type	Funktion : STRING(STRING_LENGTH)
Input	IN : STRING (Zeichenkette)
Output	STRING(STRING_LENGTH) (Zeichenkette)



URL_ENCODE wandelt reservierte Zeichen in der Zeichenkette IN in die Zeichenkette '%HH' um. In einer URL Kodierung dürfen nur die Zeichen [A..Z, a..z, 0..9, -._~] vorkommen. Anderen Zeichen können mit einem % Zeichen gefolgt vom 2 Zeichen langen Hexadezimalcode des Zeichens dargestellt werden. Das reservierte Zeichen '#' wird dabei als '%23' kodiert.

8.24. URL_TO_STRING

Type	Funktion : STRING(STRING_LENGTH)
Input	IN : STRING (Unified Resource Locator)
Output	URL (URL)



URL_TO_STRING erzeugt aus den in IN gespeicherten Daten vom Typ URL einen Unified Resource Locator als String zusammen.

Eine URL setzt sich wie folgt zusammen:

protokoll://user:passwort@domain:port/path?query#anker

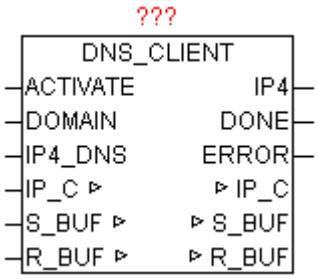
Beispiel: ftp://hugo:nono@oscat.de:1234/download/manual.html

einige Bestandteile der URL sind optional wie zum Beispiel User, Passwort, Anker, Query ...

9. Netzwerk und Kommunikation

9.1. DNS_CLIENT

Type	Funktionsbaustein
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	ACTIVATE: BOOL (Abfrage starten durch positive Flanke)
	DOMAIN: STRING (Domain Name oder IP als String)
	IP4_DNS: DWORD (IPv4 Adresse des DNS-Server)
OUTPUT	IP4: DWORD (IPv4 Adresse der angefragten Domain)
	DONE: BOOL (IP der Domain erfolgreich ermittelt)
	ERROR: DWORD (Fehlercode)



DNS_CLIENT ermittelt aus den übergebenen qualifizierten DOMAIN Namen die zugehörige IPv4 Adresse z.B. „www.oscat.de“ . Dazu wird eine DNS-Abfrage über den parametrierten DOMAIN Namen bei einen DNS-Server gemacht. Bei positiver Flanke von ACTIVATE wird die angegebene DOMAIN zwischen gespeichert, so dass diese nicht länger vorhanden sein muss. Sollte die Abfrage mehrere IP-Adressen liefern, so wird immer die mit dem höchsten Wert von TTL (Time To Live) benutzt. Als IP4_DNS kann ein beliebiger öffentlicher DNS-Server verwendet werden. Wenn die SPS hinter einen DSL-Router sitzt, kann dieser über seine Gateway-Adresse dieser als DNS-Server verwendet werden. Was letztendlich auch bei wiederkehrenden Abfragen zu schnelleren Antwortzeiten führt, da diese im Router-Cache verwaltet werden. Bei positiven Ergebnis DONE = TRUE enthält IP4 die gesuchte IP-Adresse, bis zum Start der nächsten Abfrage durch positive Flanke von ACTIVATE. Wird im DOMAIN Name eine gültige IPv4 Adresse erkannt, wird logischerweise keine DNS-Abfrage mehr

durchgeführt und gleich in konvertierter Form wieder bei IPv4 ausgegeben und DONE auf TRUE gesetzt. ERROR liefert im Fehlerfall die genaue Ursache.

Fehlercodes:

Wert				**Ursprung**	Beschreibung
B3	B2	B1	B0		
nn	**nn**	**nn**	xx	IP_CONTROL	Fehler vom Baustein IP_CONTROL
xx	xx	xx	**00**	DNS_CLIENT	No error: The request completed successfully
xx	xx	xx	**01**	DNS_CLIENT	Format error: The name server was unable to interpret the query.
xx	xx	xx	**02**	DNS_CLIENT	Server failure: The name server was unable to process this query due to a problem with the name server.
xx	xx	xx	**03**	DNS_CLIENT	Name Error: Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist
xx	xx	xx	**04**	DNS_CLIENT	Not Implemented: The name server does not support the requested kind of query
xx	xx	xx	**05**	DNS_CLIENT	Refused: The name server refuses to perform the specified operation for policy reasons
xx	xx	xx	**06**	DNS_CLIENT	YXDomain: Name Exists when it should not
xx	xx	xx	**07**	DNS_CLIENT	YXRRSet. RR: Set Exists when it should not
xx	xx	xx	**08**	DNS_CLIENT	NXRRSet. RR: Set that should exist does not
xx	xx	xx	**09**	DNS_CLIENT	Server Not Authoritative for zone
xx	xx	xx	**0A**	DNS_CLIENT	Name not contained in zone
xx	xx	xx	**FF**	DNS_CLIENT	No ip-address found

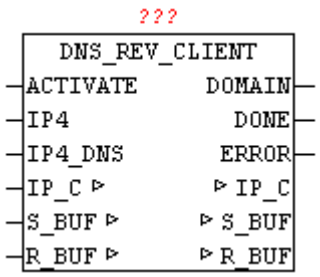
9.2. DNS_REV_CLIENT

Type Funktionsbaustein

IN_OUT IP_C : IP_C (Parametrierungsdaten)

S_BUF: NETWORK_BUFFER (Sendedaten)

	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	ACTIVATE: BOOL (Abfrage starten durch positive Flanke)
	DOMAIN: STRING (Domain Name oder IP als String)
	IP4_DNS: DWORD (IPv4 Adresse des DNS-Server)
OUTPUT	IP4: DWORD (IPv4 Adresse der angefragten Domain)
	DONE: BOOL (IP der Domain erfolgreich ermittelt)
	ERROR: DWORD (Fehlercode)



DNS_REV_CLIENT ermittelt aus den übergebenen IP-Adresse den offiziell registrierten DOMAIN Namen . Dazu wird eine Reverse DNS-Abfrage über die parametrierte IP-Adresse bei einen DNS-Server gemacht. Bei positiver Flanke von ACTIVATE wird die angegebene IP zwischen gespeichert, so dass diese nicht länger vorhanden sein muss. Sollte die Abfrage mehrere Ergebnisse liefern, so wird immer der letzte Datensatz herangezogen. Als IP4_DNS kann ein beliebiger öffentlicher DNS-Server verwendet werden. Wenn die SPS hinter einen DSL-Router sitzt, kann dieser über seine Gateway-Adresse dieser als DNS-Server verwendet werden. Was letztendlich auch bei wiederkehrenden Abfragen zu schnelleren Antwortzeiten führt, da diese im Router-Cache verwaltet werden. Bei positiven Ergebnis DONE = TRUE enthält DOMAIN den offiziell registrierten Domain-Namen, bis zum Start der nächsten Abfrage durch positive Flanke von ACTIVATE. ERROR liefert im Fehlerfall die genaue Ursache. (Siehe Fehlercodes).

Fehlercodes:

Wert				**Ursprung**	Beschreibung
B3	B2	B1	B0		
nn	**nn**	**nn**	xx	IP_CONTROL	Fehler vom Baustein IP_CONTROL
xx	xx	xx	**00**	DNS_CLIENT	No error: The request completed successfully

xx	xx	xx	**01**	DNS_CLIENT	Format error: The name server was unable to interpret the query.
xx	xx	xx	**02**	DNS_CLIENT	Server failure: The name server was unable to process this query due to a problem with the name server.
xx	xx	xx	**03**	DNS_CLIENT	Name Error: Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist
xx	xx	xx	**04**	DNS_CLIENT	Not Implemented: The name server does not support the requested kind of query
xx	xx	xx	**05**	DNS_CLIENT	Refused: The name server refuses to perform the specified operation for policy reasons
xx	xx	xx	**06**	DNS_CLIENT	YXDomain: Name Exists when it should not
xx	xx	xx	**07**	DNS_CLIENT	YXRRSet. RR: Set Exists when it should not
xx	xx	xx	**08**	DNS_CLIENT	NXRRSet. RR: Set that should exist does not
xx	xx	xx	**09**	DNS_CLIENT	Server Not Authoritative for zone
xx	xx	xx	**0A**	DNS_CLIENT	Name not contained in zone

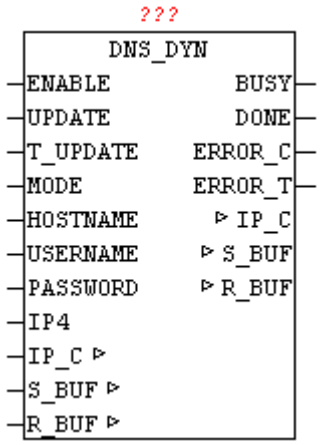
9.3. DNS_DYN

Type Funktionsbaustein

INPUT ENABLE: BOOL (Freigabe des Bausteins)
UPDATE: BOOL (Startet neue DNS-Registrierung sofort)
T_UPDATE: TIME (Wartezeit für neue DNS-Registrierung)
MODE: BYTE (Auswahl des DynDNS-Providers)
HOSTNAME: STRING(30) (eigener Domain-Name)
USERNAME: STRING(20) (Name für Registrierung)
PASSWORD: STRING(20) (Passwort für Registrierung)
IP4: DWORD (Optionale Angabe der IP-Adresse)

OUTPUT BUSY: BOOL (Baustein ist aktiv, Abfrage wird durchgeführt)
DONE: BOOL (DNS-Registrierung erfolgreich durchgeführt)
ERROR_C: DWORD (Fehlercode)
ERROR_T: BYTE (Fehlertype)

IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)



Mittels DNS_DYN können dynamische IP-Adressen als Domain-Name registriert werden. Bei vielen Internet-Providern wird bei Einwahl ins Internet eine dynamische IP-Adresse vergeben. Um von anderen Internet-Teilnehmern trotzdem auffindbar und erreichbar zu sein, ist eine der Möglichkeiten, das man seine aktuelle IP-Adresse mittels Dyn-DNS immer aktualisiert. Das Verfahren ist leider nicht standardisiert, somit ist für jeden Dyn-DNS Provider ein spezielle Lösung zu erstellen. Der Baustein kann in Verbindung mit DynDNS.org und Selfhost.de eingesetzt werden. Diese Provider bieten neben kostenpflichtigen auch kostenlose Dyn-DNS Dienste an.

Wird ENABLE auf TRUE gesetzt, dann wird der Baustein aktiv. Mittels einer positiven Flanke auf UPDATE kann jederzeit eine Aktualisierung gestartet werden. Wenn bei T_UPDATE eine Zeit angegeben ist, so wird immer nach Ablauf dieser Zeit auch eine Aktualisierung durchgeführt.

Achtung , die meisten DynDNS Provider bewerten eine zu häufiges bzw. unnötiges Updaten als Angriff, und blockieren eventuell den Account für eine gewisse Zeit.

Die Zeit T_UPDATE sollte nicht unter einer Stunde eingestellt sein. Wird der Parameter T_UPDATE nicht beschalten so wird als Updatezeit 1 Stunde angenommen. Wird kein Updaten über Zeit benötigt, dann sollte T#0ms übergeben werden.

Der Parameter MODE ermöglicht die Auswahl der DynDNS-Providers

(0 = DynDNS.org , 1 = SELFHOST.DE)

Der eigene Domainname muss bei HOSTNAME übergeben werden. Zur Sicherheit muss man beim DynDNS-Provider mittels USERNAME und PASSWORD die Authorisierungs-Daten angeben. Wird der Parameter IP4 nicht belegt, so wird vom DynDNS Provider automatisch die aktuelle WAN-IP mit der die Aktualisierung durchgeführt wird, als aktuelle Registrierung-IP angenommen. Durch Angabe von eine IP-Adresse kann aber auch eine individuelle IP-Adresse angegeben werden.

Bei fehlerfreier Durchführung wird DONE = TRUE , ansonsten wird an ERROR_C und ERROR_T der Fehlercode und der Fehlertype ausgegeben. (Siehe Fehlercodes).

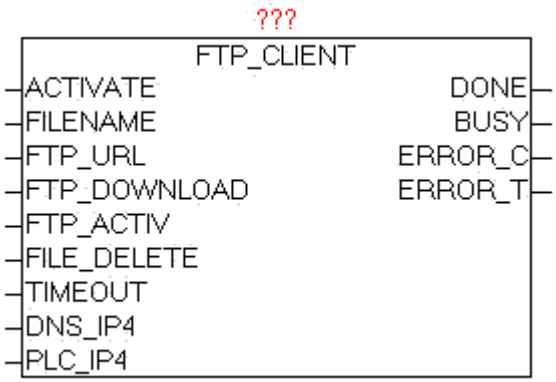
ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen
3	Der DynDNS Provider hat die Registrierung abgelehnt

9.4. FTP_CLIENT

Type	Funktionsbaustein:
INPUT	ACTIVATE : BOOL (positive Flanke startet die Abfrage)
	FILENAME : STRING (Dateipfad / Dateiname)
	FTP_URL : STRING(STRING_LENGTH) (FTP Zugriffspfad)
	FTP_DOWNLOAD : BOOL (UPLOAD = 0 / DOWNLOAD = 1)
	FTP_ACTIV : BOOL (PASSIV = 0 / ACTIV = 1)
	FILE_DELETE : BOOL (Datei nach Übertragung löschen)
	TIMEOUT : TIME (Zeitüberwachung)
	DNS_IP4 : DWORD (IP4-Adresse des DNS-Server)
	PLC_IP4 : DWORD (IP4-Adresse der eigenen Steuerung)
OUTPUT	DONE : BOOL (Transfer ohne Fehler beendet)
	BUSY : BOOL (Transfer ist aktiv)
	ERROR_C : DWORD (Fehlercode)

ERROR_T : BYTE (Fehlertype)



Der Baustein FTP_CLIENT dient zum übertragen von Dateien von der SPS zu einen FTP-Server, und zum übertragen vom FTP-Server zur SPS. Mittels positiver Flanke bei ACTIVATE wird der Übertragungsvorgang gestartet. Bei FTP_DOWNLOAD kann die Übertragungsrichtung vorgegeben werden. Der Parameter FTP_URL enthält den Namen des FTP-Server und optional den Benutzernamen und das Passwort, einen Zugriffpfad und eine zusätzliche Portnummer für den Datenkanal.Wird kein Benutzername bzw. Passwort übergeben, so versucht der Baustein sich automatisch als „Anonymous“ anzumelden. Der Parameter FTP_ACTIV bestimmt ob der FTP-Server im AKTIV oder PASSIV Modus betrieben wird. Im ACTIV Modus versucht der FTP-Server den Datenkanal zur Steuerung aufzubauen, dabei kann es jedoch durch Sicherheitssoftware, Firewall etc. zu Problemen kommen , da diese die Verbindungswunsch blockieren könnten. Hierzu müsste in der Firewall eine dementsprechende Ausnahmeregel definiert werden. Beim PASSIV Modus wird dieses Problem entschärft, da die Steuerung die Verbindung aufbaut, und somit problemlos die Firewall passieren kann. Der Steuerkanal wird immer über PORT 20 aufgebaut, und der Datenkanal standardmäßig über PORT21, dies ist aber wiederum abhängig ob ACTIV oder PASSIV Mode genutzt wird, bzw. optional eine PORT-Nummer in der FTP-URL angegeben wurde. Mit dem Parameter FILE_DELETE kann bestimmt werden, ob die Quell-Datei nach erfolgreicher Übertragung gelöscht werden soll. Dies funktioniert auf FTP als auch auf der Steuerungsseite. Bei Vorgabe von FTP-Verzeichnissen ist das Verhalten FTP-Server abhängig, ob hierbei diese schon existieren müssen, oder automatisch angelegt werden. Im Normalfall sollten diese schon vorhanden sein. Die Größe der Dateien ist an sich unbegrenzt, jedoch gibt es hier praktische Limits: Speicher auf SPS, FTP-Speicher und der Faktor Übertragungszeit. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn in der FTP-URL ein DNS-Name angegeben wird, alternativ kann in der FTP-URL auch eine IP-Adresse eingetragen werden. Bei Parameter PLC_IP4 muss die eigene IP-Adresse angegeben werden. Sollten bei der Übertragung Fehler auftreten werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY =

TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE = TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies normalerweise notwendig wäre.

Grundlagen: http://de.wikipedia.org/wiki/File_Transfer_Protocol

URL-Beispiele:

ftp://benutzername:password@servername:portnummer/verzeichnis/

ftp://benutzername:password@servername

ftp://benutzername:password@servername/verzeichnis/

ftp://servername

ftp://benutzername:password@192.168.1.1/verzeichnis/

ftp://192.168.1.1

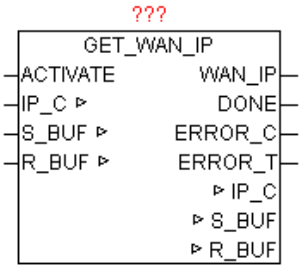
ERROR_T:

Wert	Eigenschaften
1	Störung: DNS_CLIENT Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Störung: FTP Steuerkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
3	Störung: FTP Datenkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUT ERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom FTP-Server empfangenen Response-Code. Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel: ERROR_T = 5 ERROR_C = 0x0028_00DC Ablauf-Schrittnummer 0x0028 = 40 Response-Code 0x00DC = 220

9.5. GET_WAN_IP

Type	Funktionsbaustein:
IN_OUT	IP_C : Datenstruktur 'IP_CONTROL' (Parametrierungsdaten)
	S_BUF: Datenstruktur 'NETWORK_BUFFER' (Sendedaten)
	R_BUF: Datenstruktur 'NETWORK_BUFFER' (Empfangsdaten)
INPUT	ACTIVATE: BOOL (Freigabe zur Abfrage)
OUTPUT:	WAN_IP: DWORD (Wide-Area-Network-Adresse)
	DONE: BOOL (Abfrage ohne Fehler beendet)
	ERROR_C: DWORD (Fehlercode)

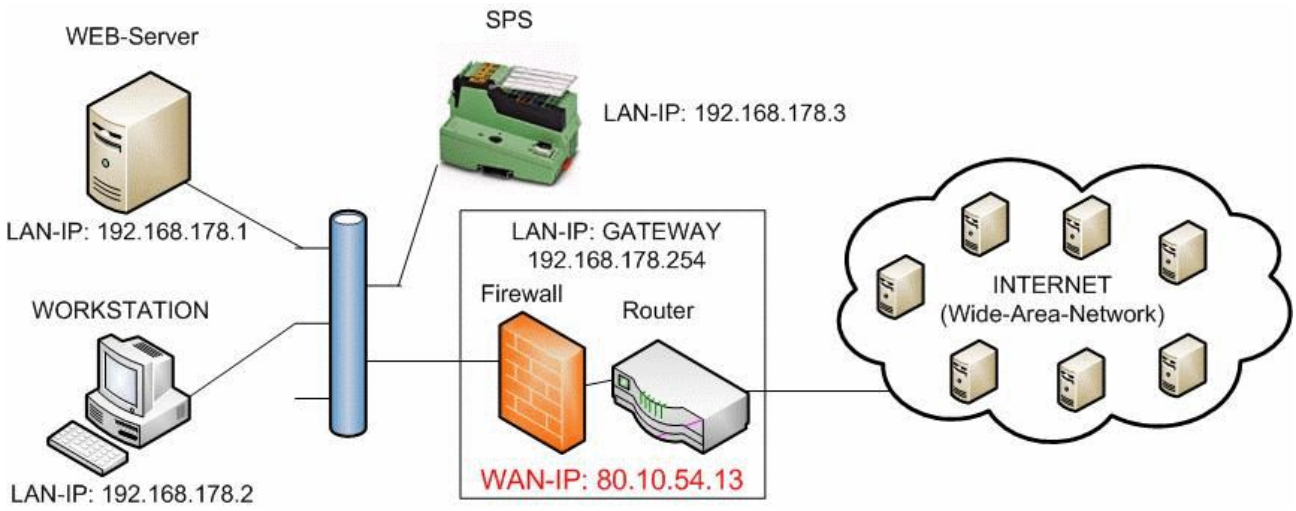
ERROR_T: BYTE (Fehlertype)



Der Baustein stellt die IP-Adresse fest, die der Internet-Routers im Wide-Area-Network (Internet) benutzt. Diese IP-Adresse ist z.B. notwendig um DynDNS Anmeldungen durchführen zu können. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben, und bei Parameter WAN_IP wird die aktuelle WAN-IP Adresse ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.

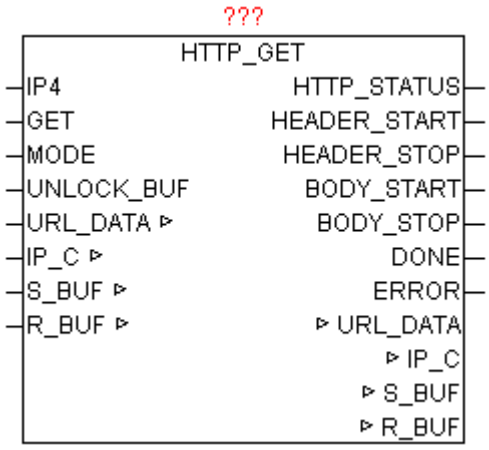
ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen



9.6. HTTP_GET

Type	Funktionsbaustein:
IN_OUT	URL_DATA : URL (Daten von STRING_TO_URL)
	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	IP4: DWORD (IP-Adresse des HTTP-Servers)
	GET: BOOL (Startet die HTTP Abfrage)
	MODE: BYTE (Version der HTTP-GET Abfrage)
	UNLOCK_BUF: BOOL (Freigabe des Empfangs-Datenbuffers)
OUTPUT	HTTP_STATUS: STRING (ermittelter HTTP Statuscode)
	HTTP_START: UINT (Start-Position des Message-Headers)
	HTTP_STOP: UINT (Stopp-Position des Message-Headers)
	BODY_START: UINT (Start-Position des Message-Body)
	BODY_STOP: UINT (Stopp-Position des Message-Body)
	DONE: BOOL (Aufgabe ohne Fehler durchgeführt)
	ERROR: DWORD (Fehlercode)



HTTP_GET führt nach positiver Flanke von GET ein GET-Kommando auf einem HTTP-Server durch. Mittels MODE kann die HTTP Protokollversion vorgegeben werden. Die gewünschte URL (Web-Link) muss vorab in der URL_DATA Struktur fertig aufbereitet vorliegen. Die vollständige URL sollte darum vor dem Bausteinaufruf mittels „STRING_TO_URL“ aufbereitet werden. Nach erfolgreicher Abfrage wird DONE = TRUE ausgegeben, und die Parameter HTTP_START und HTTP_STOP zeigen auf den Datenbereich in dem die Message-Header Daten zur weiteren Bearbeitung und Auswer-

tung vorzufinden sind. Im Normalfall ist auch ein Message-Body vorhanden, der wiederum mittels BODY_START und BODY_STOP übermittelt wird. Weiters wird auf HTTP_STATUS der zurückgemeldete HTTP-Statuscode als String ausgegeben. Eine der Schwierigkeiten beim HTTP Empfang ist das erkennen vom Ende des Datenstream. Dabei werden vom Baustein mehrere Strategien verfolgt. Bei Anwendung von HTTP/1.0 wird das Ende des Datenempfangs über einen Verbindungsabbau seitens des Host erkannt. Weiters wird immer geprüft ob sich im Header die Info „Content-Length“ befindet, damit kann dann eindeutig erkannt werden ob alle Daten empfangen wurden. Trifft keines der vorigen Varianten zu, so wird über einen einfachen Receive-Timeout Error das Ende der Datenübertragung festgestellt und beendet. Einziger Nachteil dabei ist, das dies je nach eingestellte Timeout Zeit mitunter länger dauert als gewünscht. Darum ist es nicht schlecht wenn eine vernünftiger Timeout-Wert am IP_CONTROL gewählt wird. ERROR liefert im Fehlerfall die genaue Ursache (Siehe Baustein IP_CONTROL)

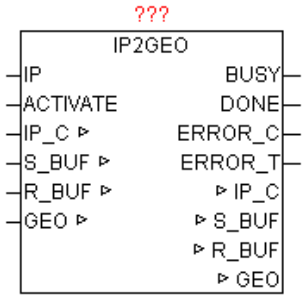
Folgende MODE können verwendet werden:

Mode	Protokollversion	Eigenschaften
0	HTTP/1.0	Der Host beendet selbstständig nach Übertragung der Daten die TCP-Verbindung
1	HTTP/1.0	Durch Anwendung von „Connection: Keep-Alive“ wird der Host angewiesen eine persistente Verbindung zu verwenden. Der Client sollte nach Ende der Aktivitäten die Verbindung wieder abbauen.
2	HTTP/1.1	Der Host verwendet eine persistente Verbindung und muss von Client beendet werden.
3	HTTP/1.1	Durch Anwendung von „Connection: Close“ wird der Host angewiesen nach Übertragung der Daten die TCP-Verbindung zu beenden.

9.7. IP2GEO

Type Funktionsbaustein:

IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
	GEO: IP2GEO (Geographische Daten)
INPUT	ACTIVATE: BOOL (Freigabe zur Abfrage)
OUTPUT	BUSY: BOOL (Abfrage ist aktiv)
	DONE: BOOL (Abfrage ohne Fehler beendet)
	ERROR_C: DWORD (Fehlercode)
	ERROR_T: BYTE (Fehlertype)



Der Baustein liefert aufgrund der Wide-Area-Network IP-Adresse die Geographischen Informationen des Internet-Zuganges. Da die WAN-IP-Adressen weltweit registriert sind, lässt sich somit annähernd die Geographische Position der SPS bestimmen. Sollte der Zugang über einen Proxy-Server laufen, so wird die Geographische Position von diesen ermittelt und nicht von der SPS. Normalerweise ist der aber in unmittelbarer Nähe der SPS, und somit die Abweichung nicht relevant. Somit ergeben sich ermittelte Positionen die nur ein paar Kilometer von der wirklichen Position abweichen, und relativ genau sind.

Wird am Parameter „IP“ keine IP-Adresse angegeben, so wird automatisch die aktuelle WAN-IP herangezogen, andernfalls werden die Informationen der parametrierten IP geliefert. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet. Solange die Abfrage nicht beendet ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben, und bei Parameter WAN_IP wird die aktuelle WAN-IP Adresse ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.

ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen

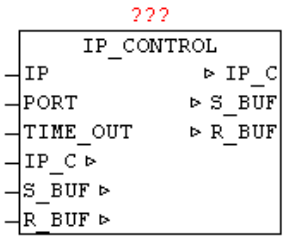
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

Der „COUNTRY_CODE ist nach „ISO 3166 Country-Code ALPHA-2“ kodiert.
http://www.iso.org/iso/english_country_names_and_code_elements
http://de.wikipedia.org/wiki/ISO-3166-1-Kodierliste

Der „REGION_CODE“ ist nach „FIPS Region Code“ kodiert.
http://en.wikipedia.org/wiki/List_of_FIPS_region_codes

9.8. IP_CONTROL

Type	Funktionsbaustein
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	IP: DWORD (kodierte IP Adresse als Standardadresse)
	PORT: WORD (Portnummer der IP-Adresse)
	TIME_OUT: TIME (Überwachungszeit)



Verfügbare Plattformen und damit verbundene Abhängigkeiten

CoDeSys:

Benötigt die Bibliothek „SysLibSockets.lib“

Lauffähig auf

WAGO 750-841
CoDeSys SP PLCWinNT V2.4

und kompatible Plattformen

PCWORX:

Keine Bibliothek notwendig

Lauffähig auf allen Steuerungen mit Ethernet ab Firmware >= 3.5x

BECKHOFF:

Erfordert die Installation des„TwinCAT TCP/IP Connection Server“
Benötigt somit die Bibliothek „TcpIp.Lib“
(Standard.Lib; TcBase.Lib; TcSystem.Lib werden danach automatisch eingebunden)

Programmierumgebung:
NT4, W2K, XP, Xpe;
TwinCAT System Version 2.8 oder höher;
TwinCAT Installation Level: TwinCAT PLC oder höher;
Zielplattform:
TwinCAT SPS-Laufzeitsystem Version 2.8 oder höher.
PC or CX (x86)
TwinCAT TCP/IP Connection Server v1.0.0.0 oder höher;
NT4, W2K, XP, XPe, CE (image v1.75 oder höher);
CX (ARM)
TwinCAT TCP/IP Connection Server v1.0.0.44 oder höher;
CE (image v2.13 oder höher);

Der IP_CONTROL ermöglicht die Hersteller und Plattform neutrale Nutzung der Ethernet-Kommunikation. Um die vielen verschiedene Schnittstellen der Steuerungslieferanten zu vereinen, wird der IP_CONTROL als Adapter „WRAPPER“ eingesetzt. Mit diesem Baustein können UDP und TCP sowie aktive und passive Verbindungen gehandhabt werden. Bei manchen Kleinsteuerungen ist auch die Anzahl an gleichzeitige geöffneten Sockets sehr begrenzt, darum unterstützt dieser Baustein auch das Sharing von Sockets. Durch eine integrierte automatische Koordinierung der Zugriffe ist es möglich das sich viele Client-Bausteine einen Socket teilen. Hierbei wird automatisch bei Zugriff erkannt ob ein Client andere Verbindungsparameter als sein Vorgänger benutzt. Eine bestehende Verbindung wird gegebenenfalls automatisch beendet und mit den neuen Verbindungsparametern wieder aufgebaut. Die Art der Verbindung kann mittels C_MODE eingestellt werden (Siehe Tabelle). Über C_PORT wird die gewünschte Port-Nummer vorgegeben, und mittels C_IP die IP v4 Adresse. Über C_STATE kann festgestellt werden ob die Verbindung auf – abgebaut ist , bzw. die negative und positive Flanke bei Änderung des Zustandes. Mittels C_ENABLE wird die Verbindung freigeben (aufgebaut) bzw. gesperrt (abgebaut). Das Daten senden und empfangen funktioniert unabhängig voneinander, das heißt es ist auch asynchrones Senden und Empfangen

möglich wie z.B. bei Telnet. Bei Anwendungen bei denen nur gesendet wird, und kein Datenempfang erwartet wird, muss R_OBSERVE = FALSE sein, damit kein Timeout beim Empfang auftritt. Beim Beginn von Sende und Empfangsaktivitäten wird TIME_RESET vom Anwender einmalig auf TRUE gesetzt, damit alle Timeout mit einen definierten Startwert neu beginnen. Dies ist aufgrund der Sharing Funktionalität erforderlich, denn hier können aufgebaute Verbindungen bestehen bleiben und die Zugriffsrechte werden dabei nur weitergereicht. Der Parameter IP dient als mögliche Default-IP-Adresse. Um nicht mehrmals die gleiche IP-Adresse parametrieren zu müssen, kann eine Default - IP verwendet werden. Ein möglicher Einsatz wäre die Angabe der DNS-Server-Adresse. Wenn der Baustein als C_IP die IP 0.0.0.0 erkennt, so wird automatisch die parametrierte IP Adresse verwendet. Das gleiche Verhalten besteht beim Parameter Port. Wird bei C_PORT der Port 0 erkannt so wird die am Baustein parametrierte Portnummer verwendet. Der Fehlercode ERROR setzt sich aus mehreren Teilen zusammen (Siehe ERROR Tabelle). Mittels TIMEOUT kann die allgemeine Überwachungszeit vorgeben werden. Dieser Zeitwert wird für Verbindungsaufbau, Daten senden und Daten empfangen jeweils unabhängig voneinander verwendet. Der übergebene TIMEOUT Zeitwert wird automatisch auf 200ms Minimum begrenzt. Somit kann dieser Parameter auch frei bleiben.

Wenn bei freigegebener Verbindung in den Sende buffer die Daten und eine Datenlänge eingetragen werden, wird automatisch der Datenblock versendet. Bei Datenempfang werden die Daten immer wieder zu den schon bestehenden Daten im Buffer angehängt. Durch eintragen von SIZE = 0 wird der Empfangs-Datenzeiger wieder zurückgesetzt, und die nächsten empfangenen Daten werden wieder ab Position 0 abgelegt.

Der Baustein unterstützt das Blocken der Datentelegramme, das heißt der S_BUF bzw. R_BUF kann beliebig groß sein. Einzelne empfangene Daten-Telegramme werden im R_BUF in Form eines Stream gesammelt. Genauso wird beim Senden von Daten verfahren. Die Daten im S_BUF werden in zulässiger Blockgröße einzeln als Stream versendet.

Anwendungs-Beispiel:

```
    CASE state OF
00: (* auf Freigabe warten *)
  IF FREIGABE THEN
    state := 10;
    IP_STATE := 1; (* Anmelden *)
  END_IF;
10: (* Warten auf Zugriffsfreigabe zum Verbindung aufbauen und Datensenden *)
  IF IP_STATE = 3 THEN (* Zugriff erlaubt ? *)
    (* IP Datenverkehr einrichten *)
```

```
    IP_C.C_PORT := 123; (* Port-Nummer eintragen *)
    IP_C.C_IP := IP4; (* IP eintragen *)
    IP_C.C_MODE := 1; (* Mode: UDP+AKTIV+PORT+IP *)
    IP_C.C_ENABLE := TRUE; (* Verbindungsaufbau freigeben *)
    IP_C.TIME_RESET := TRUE; (* Zeitüberwachung rücksetzen *)
    IP_C.R_OBSERVE := TRUE; (* Datenempfang überwachen *)
    R_BUF.SIZE := 0; (* Empfangslänge rücksetzen *)

    (* Sendedaten eintragen *)
    S_BUF.BUFFER[0] := BYTE#16#1B;
    (* usw.... *)
    S_BUF.SIZE := xx; (* Sendelänge eintragen *)
    state := 30;
30:
  IF IP_C.ERROR <> 0 THEN
    (* Fehlerauswertung durchführen *)
    ELSIF S_BUF.SIZE = 0 AND R_BUF.SIZE >= xxx THEN
     (* Empfangene Daten auswerten *)

     (* Abmelden - Zugriff wieder für andere freigeben *)
     IP_STATE := BYTE#4;
     state := 00; (* Ablauf beenden *)
  END_IF;
END_CASE;

(* IP_FIFO Zyklisch aufrufen *)
IP_FIFO(FIFO:=IP_C.FIFO,STATE:=IP_STATE,ID:=IP_ID);
IP_C.FIFO:=IP_FIFO.FIFO;
IP_STATE := IP_FIFO.STATE;
IP_ID:=IP_FIFO.ID;
```

Folgende C_MODE können verwendet werden:

TYP	TCP / UDP	Aktiv / Passiv	Port-Nummer erforderlich	IP-Adresse erforderlich
0	TCP	Aktiv	Ja	Ja
1	UDP	Aktiv	Ja	Ja
2	TCP	Passiv	Ja	**Ja** (Adresse des aktiven Partners)

3	UDP	Passiv	Ja	**Ja** (Adresse des aktiven Partners)
4	TCP	Passiv	Ja	**Nein** (beliebiger aktiver Partner wird akzeptiert)
5	UDP	Passiv	Ja	**Nein** (beliebiger aktiver Partner wird akzeptiert)

C_STATE:

Wert	**Zustandsmeldung**
0	Verbindung ist abgebaut
1	Verbindung wurde abgebaut (negative Flanke) Wert ist nur für einen Zyklus vorhanden, dann wird 0 ausgegeben
254	Verbindung wurde aufgebaut (positive Flanke) Wert ist für einen Zyklus vorhanden,dann wird 255 ausgegeben
255	Verbindung ist aufgebaut
<127	Abfrage ob Verbindung abgebaut ist
>127	Abfrage ob Verbindung aufgebaut ist

ERROR:

DWORD				**Meldungstype**	Beschreibung
B3	B2	B1	B0		
00	xx	xx	xx	Verbindungsaufbau	Wert 00 - Keine Fehler vorhanden
nn	xx	xx	xx	Verbindungsaufbau	Wert 01 - 252 System spezifischer Fehler
FD	xx	xx	xx	Verbindungsaufbau	Wert 253 – Verbindung von Remote beendet
FF	xx	xx	xx	Verbindungsaufbau	Wert 255 - Timeout Fehler
xx	**00**	xx	xx	Daten senden	Wert 00 - Keine Fehler vorhanden
xx	**nn**	xx	xx	Daten senden	Wert 01 - 252 System spezifischer Fehler
xx	**FF**	xx	xx	Daten senden	Wert 255 - Timeout Fehler
xx	xx	**00**	xx	Daten empfangen	Wert 00 - Keine Fehler vorhanden
xx	xx	**nn**	xx	Daten empfangen	Wert 01 – 252 System spezifischer Fehler
xx	xx	**FF**	xx	Daten empfangen	Wert 255 - Timeout Fehler
xx	xx	**FE**	xx	Daten empfangen	Wert 254 – Empfangsbuffer ist voll (Überlauf)

					(Buffer-Size wird automatisch auf 0 gesetzt)
xx	xx	xx	**nn**	Applikation-Error	Bei IP_CONTROL immer 00. ERROR wird von der Client-Applikation 1:1 übernommen und gegebenenfalls wird an dieser Stelle ein Applikation-Error eingetragen. Diese Fehlercode wird aber nur von den Client-Bausteinen eingetragen.

System spezifische Fehler: (PCWORX / MULTIPROG)

Wert	**Zustandsmeldung**
0x00	Kein Fehler aufgetreten.
0x01	Erstellen mindestens einer Task ist fehlgeschlagen.
0x02	Initialisierung des Socket-Interfaces fehlgeschlagen (nur WinNT).
0x03	Dynamischer Speicher konnte nicht reserviert werden.
0x04	FB kann nicht initialisiert werden, da beim Starten der asynchronen Kommunikations-Tasks ein Fehler aufgetreten ist.
0x10	Socket-Initialisierung fehlgeschlagen.
0x11	Fehler beim Senden einer Nachricht.
0x12	Fehler beim Empfang einer Nachricht.
0x13	Unbekannter Service-Code im Telegramm-Header.
0x21	Ungültiger Zustandsübergang beim Verbindungsaufbau.
0x30	Keine freien Kanäle mehr verfügbar.
0x31	Die Verbindung wurde abgebrochen.
0x33	Allgemeiner Timeout, Empfänger antwortet nicht mehr bzw. Sender hat Übertragung nicht beendet.
0x34	Verbindungswunsch wurde negativ quittiert.
0x35	Empfänger hat Sendung nicht bestätigt, evtl. ist Empfänger überlastet (Übertragung wiederholen).
0x40	Partner-String ist zu lang (255 Zeichen maximal).
0x41	Die angegebene IP-Adresse ist nicht gültig oder konnte nicht richtig interpretiert werden.
0x42	Nicht zulässige Port-Nummer.
0x45	Unbekannter Parameter an Eingang "PARTNER".
0x50	Sendeversuch auf ungültiger Verbindung (Sender oder Empfänger) .

0x53	Alle verfügbaren Verbindungen sind belegt.
0x61	Neg. Confirmation des Empfängers. Es wurde eine falsche Sequenznummer verwendet.
0x62	Datentyp von Sender und Empfänger sind ungleich.
0x63	Empfänger ist im Augenblick nicht empfangsbereit (mgl. Ursache: Empfänger ist disabled oder befindet sich gerade in der Datenübernahme (NDR=TRUE).
0x64	Es wurde kein Empfangsbaustein mit der entsprechenden R_ID gefunden.
0x65	Eine andere Baustein-Instanz arbeitet bereits auf dieser Verbindung.
0x70	Partner-Steuerung wurde nicht als Time-Server konfiguriert.

System spezifischer Fehler: (CoDeSys)

0x00	Kein Fehler aufgetreten.
0x01	SysSockCreate nicht erfolgreich ausgeführt
0x02	SysSockBind nicht erfolgreich ausgeführt
0x03	SysSockListen nicht erfolgreich ausgeführt

System spezifischer Fehler: (BECKHOFF)

0x00	Kein Fehler aufgetreten.
0x01	SocketUdpCreate nicht erfolgreich ausgeführt
0x03	SocketListen nicht erfolgreich ausgeführt
0x04	SocketAccept nicht erfolgreich ausgeführt

Wichtige Hinweise:

Zusatzinformation für Beckhoff

Es können mehrere Netzwerkadapter in einem PC existieren. Die globale Variable sLocalHost bestimmt den Netzwerkadapter der benutzt werden soll. Wenn Sie die globale sLocalHost-Variable ignorieren (Leerstring), dann wird von dem TCP/IP Connection Server der Default-Netzwerkadapter benutzt. Es ist meistens der erste Netzwerkadapter aus der Liste der Netzwerkadapter in der Systemsteuerung.

1. Wenn sie als sLocalHost einen Leerstring angegeben haben und der PC vom Netzwerk getrennt wurde, dann öffnet das System einen neuen Socket unter der Software-Loopback-IP-Adresse: '127.0.0.1'.

2. Wenn im PC zwei oder mehr Netzwerkadapter vorhanden sind und Sie als sLocalHost einen Leerstring angegeben haben, der Default-Netzwerkadapter aber vom Netzwerk getrennt wurde, dann wird der neue Socket unter der der IP-Adresse des zweiten Netzwerkadapters geöffnet.

3. Um das Öffnen der Sockets unter einer anderen IP-Adresse zu verhindern können Sie die sLocalHost-Adresse explizit angeben.

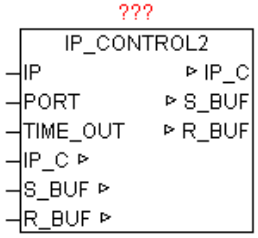
Die globale Variable sSrvNetId beinhaltet die Netzwerk Adresse des TwinCAT TCP/IP Connection Server (z.B. '1.1.1.2.7.1'). Für den lokalen Computer (Default), muss ein Leerstring angegeben werden.

Zusatzinformation für Codesys

Mit der globale Variable SYSLIBSOCKETS_OPTION kann mit dem Wert 1 auf einen Kompatibilitätsmodus umgeschaltet werden. Dadurch wird beim Verbindungsaufbau intern die Funktion SYSSOCKSELECT verwendet. Dies ist z.B. bei die Runtime „CoDeSys SP PLCWinNT V2.x“ notwendig bzw. bei allen ähnlichen Systemen.

9.9. IP_CONTROL2

Type	Funktionsbaustein
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER_SHORT (Sendedaten)
	R_BUF: NETWORK_BUFFER_SHORT (Empfangsdaten)
INPUT	IP: DWORD (kodierte IP Adresse als Standardadresse)
	PORT: WORD (Portnummer der IP-Adresse)
	TIME_OUT: TIME (Überwachungszeit)



Verfügbare Plattformen und damit verbundene Abhängigkeiten

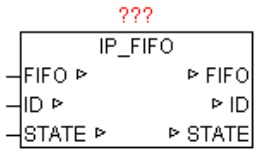
(Siehe Baustein IP_CONTROL)

Der Baustein hat prinzipiell die gleiche Funktionalität wie IP_CONTROL. Jedoch sind S_BUF und R_BUF vom TYP 'NETWORK_BUFFER_SHORT' (Siehe allgemeine Beschreibung IP_CONTROL).

Es wird bei IP_CONTROL2 kein Blocken der Daten unterstützt. Die maximale Datengröße beim Senden und Empfangen ist abhängig von der Hardware-Plattform und bewegt sich im Bereich < 1500 Bytes. Dieser Baustein kann immer dann eingesetzt werden wenn kein Data-Stream Modus notwendig ist. Primärer Vorteil hierbei ist, das weniger Buffer-Speicher belegt wird, und keine Daten zwischen internen und externen Daten Buffer kopiert werden muss. Somit ist der Baustein bezüglich Speicherverbrauch und Systembelastung sparsamer.

9.10. IP_FIFO

Type	Funktionsbaustein:
IN_OUT	FIFO : IP_FIFO_DATA (IP-FIFO Verwaltungsdaten)
	ID: BYTE (aktuelle vom IP_FIFO-Baustein vergebene ID)
	STATE: BYTE (Steuerbefehle und Statusmeldungen)



Dieser Baustein dient in Kombination mit IP_CONTROL zur Ressourcen-Verwaltung. Damit ist es möglich das Client-Applikation die alleinigen Zugriffsrechte anfordern und auch wieder abgeben können. Durch das FIFO wird gewährleistet das jeder Teilnehmer gleich oft den Ressourcen-Zugriff zugeteilt bekommt.

Bei ersten Aufruf des Bausteins wird automatisch eine neue eindeutige Applikation-ID vergeben, mittels der dann die Verwaltung im FIFO durchgeführt wird. Der Parameter STATE wird von der Applikation als auch vom IP_FIFO Baustein verändert. Jede Applikation kann sich in der Standardeinstellung nur einmal im FIFO eintragen.

STATE:

Wert	Zustandsmeldung
0	Keine Aktion
1	Zugriffsrecht anfordern
2	Zugriffsrecht anfordern wurde in FIFO übernommen
3	Zugriffsrecht erhalten (Ressourcen-Zugriff erlaubt)
4	Zugriffsrecht abgeben
5	Zugriffsrecht wurde aus FIFO wieder entfernt

Ablauf:

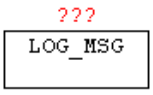
1. Applikation setzt STATE auf 1
2. Zugriffsrechte werden angefordert, während dessen ist STATE=2
3. wenn Ressource frei ist, und Zugriffsrechte vorhanden sind, dann ist STATE=3
4. Wenn die Applikation die Ressource bzw. den Zugriff nicht mehr benötigt,wird von der Applikation STATE auf 4 gesetzt. Danach gibt IP_FIFO die Ressource wieder frei und setzt STATE auf 5.
5. Vorgang wiederholt sich (gleicher oder anderer Applikation)

Anwendungsbeispiel ist beim Baustein IP_CONTROL zu finden !

9.11. LOG_MSG

Type Funktionsbaustein:

IN_OUT LOG_CL: LOG_CONTROL (LOG-Daten)



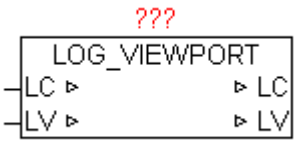
Mit LOG_MSG können Meldungen (STRINGS) in einen RING-BUFFER abgelegt werden. Die Meldungen können mit Zusatzparameter versehen werden wie Frontcolor und Backcolor für die Ausgabe auf TELNET sowie ein Eintragsfilter durch vorgabe eines Nachrichten-Level. Ist der Level der Nachricht grösser als der Vorgabe-Log-Level , so wird diese Nachricht verworfen. Weiters kann mit Enable auch das Logging allgemein

deaktiviert werden. Somit ist es kein Problem viele Meldungen pro SPS-Zyklus zu archivieren. Der Nachrichtenbuffer kann z.B. mit dem Baustein TELNET_LOG auf einen TELNET-CLIENT ausgegeben werden. Details zur Schnittstelle sind in der nachfolgenden Tabelle ersichtlich.

Wenn aus verschiedenen Bausteininstanzen Meldungen in den selben LOG_BUFFER eingetragen werden sollen, so muss die „LOG_CL“ Datenstruktur GLOBAL angelegt werden.

9.12. LOG_VIEWPORT

Type	Funktionsbaustein
IN_OUT	LC : LOG_CONTROL
	LV : us_LOG_VIEWPORT



Der Baustein LOG_VIEWPORT dient zum erzeugen einer Indexliste der LOG_CONTROL Meldungen die sich aktuell in der virtuellen Ansicht befinden. Um sich innerhalb der Meldungsliste zu bewegen (scrollen) kann über LV.MOVE_TO_X ein Scroll-offset angegeben werden. Ein positiver Wert scrollt in richtig neuere Meldungen und ein negativer Wert in richtig der älteren Meldungen. Die Anzahl der Zeilen der Meldungsliste der virtuellen Ansicht wird über LV.COUNT vorgegeben. Die in der aktuellen virtuellen Ansicht sich befindenden Meldungen werden in LV.LINE_ARRAY[x] abgelegt, und stehen für die weitere Verarbeitung zu Verfügung. Eine Veränderung der Meldungsliste wird immer mit LV.UPDATE := TRUE signalisiert, und muss vom Anwender wieder rückgesetzt werden.

Folgende LV.MOVE_TO_X Werte erzeugen ein spezielles Verhalten.

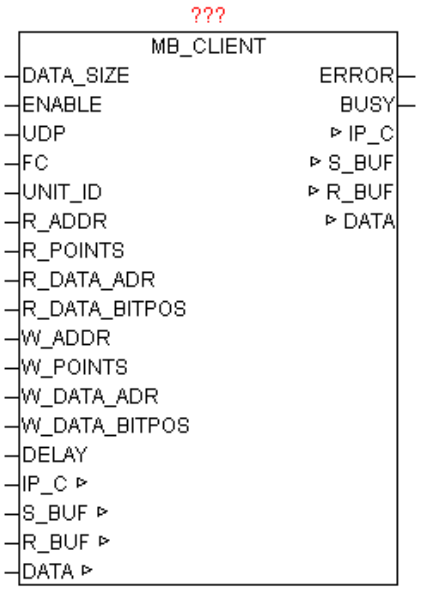
+30000 = älteste Meldungen anzeigen (Anfang des Ringbuffer)

+30001 = neueste Meldungen anzeigen (Ende des Ringbuffer)

+30002 = eine ganze Seite in Richtig neuere Meldungen scrollen
+30003 = eine ganze Seite in Richtig ältere Meldungen scrollen

9.13. MB_CLIENT (OPEN-MODBUS)

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER_SHORT (Sendedaten)
	R_BUF: NETWORK_BUFFER_SHORT (Empfangsdaten)
	DATA: ARRAY [0..255] OF WORD (MODBUS-Register)
INPUT	DATA_SIZE: INT (Anzahl der Datenwörter in Struktur MB_DATA)
	ENABLE: BOOL (Freigabe)
	UDP: BOOL (Vorwahl TCP / UDP, TRUE = UDP)
	FC: INT (Funktionsnummer)
	UNIT_ID: BYTE (Geräte-Identifikationsnummer)
	R_ADDR: INT (Lesebefehl: MODBUS Datenpunkt Adresse)
	R_POINTS: INT (Lesebefehl: MODBUS Anzahl der Datenpunkte)
	R_DATA_ADR: INT (Lesebefehl: DATA Datenpunkt Adresse)
	R_DATA_BITPOS: INT (Lesebefehl: DATA Datenpunkt Bitpos.)
	W_ADDR: INT (Lesebefehl: MODBUS Datenpunkt Adresse)
	W_POINTS: INT (Lesebefehl: MODBUS Anzahl der Datenpunkte)
	W_DATA_ADR: INT (Lesebefehl: DATA Datenpunkt Adresse)
	W_DATA_BITPOS: INT (Lesebefehl: DATA Datenpunkt Bitpos.)
	DELAY: TIME (Wiederholungszeit)
OUTPUT	ERROR: DWORD (Fehlercode)
	BUSY: BOOL (Baustein ist aktiv)



Der Baustein ermöglicht den Zugriff auf Ethernet-Geräte die MODBUS-TCP bzw. MODBUS-UDP unterstützten, bzw. MODBUS RS232/485 Geräte die über Ethernet-Modbus-Gateway angebunden sind. Es werden Kommandos aus den Klassen 0,1,2 unterstützt. Die Parameter IP_C, S_BUF, R_BUF bilden hierbei die Schnittstelle zum IP_CONTROL Baustein der hier zur Abwicklung bzw. Koordinierung benutzt wird. Die gewünschte IP-Adresse und Port-Nummer (Standard für MODBUS ist 502) muss am IP_CONTROL zentral angegeben werden. Die DATA-Struktur ist als WORD-Array ausgeführt und beinhaltet die MODBUS-Daten für Lesen und Schreiben. Mittels DATA_SIZE wird die Größe des WORD_ARRAY angegeben. Durch ENABLE wird der Baustein freigegeben, und bei Wegnahme der Freigabe wird eine eventuell aktive Abfrage noch beendet. Bei Geräten die MODBUS-UDP unterstützen kann mittels UDP=TRUE dieser Modus aktiviert werden. Der Parameter UNIT_ID muss nur bei Nutzung eines Ethernet-Modbus-Gateway angegeben werden. Die gewünschte Funktion wird mittels FC angegeben (Siehe Funktionscode-Tabelle). Je nach Funktion müssen die R_xxx und W_xxx Parameter mit Daten versorgt werden. Durch Angabe DELAY kann die die Wiederholungszeit vorgegeben werden. Wird keine Zeit angeben so wird versucht so oft wie möglich das Kommando auszuführen. Durch die integrierte Zugriffsverwaltung ist automatisch sichergestellt, das andere Baustein-Instanzen auch an die Reihe kommen. Eine negative Befehlsausführung wird mittels ERROR gemeldet (Siehe ERROR-Tabelle).Wenn der Baustein aktiv eine Abfrage durchführt, dann wird während dieser Zeit BUSY=TRUE ausgegeben.

Unterstützte Funktionscodes und verwendete Parameter:

Funktionscode	Bit Zugriff	16 Bit Zugriff (Register)	Gruppe	Funktion Beschreibung	R_ADDR	R_POINTS	R_DATA_ADR	R_DATA_BITPOS	W_ADDR	W_POINTS	W_DATA_ADR	W_DATA_BITPOS
1	x		Coils	Read Coils	x	x	x	x				
2	x		Input Discrete	Read Discrete Inputs	x	x	x	x				
3		x	Holding Register	Read Holding Registers	x	x	x					
4		x	Input Register	Read Input Register	x		x					
5	x		Coils	Write Single Coil					x		x	x
6		x	Holding Register	Write Single Register					x		x	
15	x		Coils	Write Multiple Coils					x	x	x	x
16		x	Holding Register	Write Multiple Register					x	x	x	
22		x	Holding Register	Mask Write Register					x		x	
23		x	Holding Register	Read/Write Multiple Register	x	x	x		x	x	x	

ERROR:

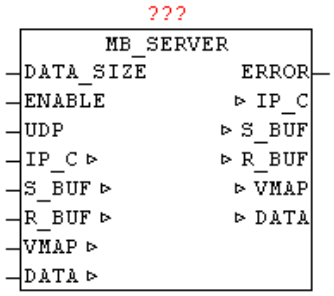
Wert				**Ursprung**	Beschreibung
B3	B2	B1	B0		
nn	**nn**	**nn**	xx	IP_CONTROL	Fehler vom Baustein IP_CONTROL
xx	xx	xx	**00**	MB_CLIENT	No Error
xx	xx	xx	**01**	MB_CLIENT	ILLEGAL FUNCTION: The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
xx	xx	xx	**02**	MB_CLIENT	ILLEGAL DATA ADDRESS: The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100

					registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 “Illegal Data Address” since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100.
xx	xx	xx	**03**	MB_CLIENT	ILLEGAL DATA VALUE: A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
xx	xx	xx	**04**	MB_CLIENT	SLAVE DEVICE FAILURE: An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
xx	xx	xx	**05**	MB_CLIENT	ACKNOWLEDGE: Specialized use in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a Poll Program Complete message to determine if processing is completed.
xx	xx	xx	**06**	MB_CLIENT	SLAVE DEVICE BUSY: Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long–duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
xx	xx	xx	**8**	MB_CLIENT	MEMORY PARITY ERROR: Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check.The server (or slave) attempted to read record file, but detected a parity error in the memory.The client (or master) can retry the request, but service may be required on the server (or slave) device.
xx	xx	xx	**0A**	MB_CLIENT	GATEWAY PATH UNAVAILABLE: Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
xx	xx	xx	**0B**	MB_CLIENT	GATEWAY TARGET DEVICE FAILED TO RESPOND: Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the

					device is not present on the network.

9.14. MB_SERVER (OPEN-MODBUS)

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER_SHORT (Sendedaten)
	R_BUF: NETWORK_BUFFER_SHORT (Empfangsdaten)
	VMAP: ARRAY [1..10] OF VMAP_DATA (Virtuelle Adresstabellen)
	DATA: ARRAY [0..255] OF WORD (MODBUS-Register)
INPUT	DATA_SIZE: INT (Anzahl der Datenwörter in DATA)
	ENABLE: BOOL (Freigabe)
	UDP: BOOL (Vorwahl TCP / UDP, TRUE = UDP)
OUTPUT	ERROR: DWORD (Fehlercode)



Der Baustein ermöglicht den Zugriff von Extern auf lokale MODBUS-Datentabellen über Ethernet . Es werden Kommandos der Klassen 0,1,2 unterstützt. Die Parameter IP_C, S_BUF, R_BUF bilden hierbei die Schnittstelle zum IP_CONTROL Baustein der hier zur Abwicklung bzw. Koordinierung benutzt wird. Die gewünschte Port-Nummer (Standard für MODBUS ist 502) muss am IP_CONTROL zentral angegeben werden. Die IP-Adresse wird am IP_CONTROL nicht benötigt, da dieser hier im Modus PASSIV agiert. Die DATA-Struktur ist als WORD-Array ausgeführt und beinhaltet die MODBUS-Daten. Mittels DATA_SIZE wird die Größe von DATA angegeben. Durch ENABLE wird der Baustein freigegeben, und bei Wegnahme der Freigabe wird eine eventuell aktive Abfrage noch beendet. Bei Geräten die MODBUS-UDP unterstützen kann mittels UDP=TRUE dieser Modus aktiviert werden. Eine negative Befehlsausführung wird mittels ERROR gemeldet (Siehe ERROR-Tabelle).

Mittels Einträgen in der Datenstruktur VMAP können virtuelle Datenbereiche angelegt werden, und der Zugriff für bestimmte Funktionscodes und Datenbereiche parametriert werden. Somit ist es sehr einfach möglich virtuelle Adressbereiche in einen zusammenhängenden Datenblock (DATA) abzubilden, oder Datenbereiche mit einen Schreibschutz zu versehen, oder Bereiche die mit Ausgangsperipherie gekoppelt sind mit einer Watchdog zu versehen.

Die Handhabung der VMAP-Daten ist beim Baustein MB_VMAP näher beschrieben.

ERROR:

Wert				**Ursprung**	Beschreibung
B3	B2	B1	B0		
nn	**nn**	**nn**	xx	IP_CONTROL	Fehler vom Baustein IP_CONTROL
xx	xx	xx	**00**	MB_SERVER	NO ERROR:
xx	xx	xx	**01**	MB_SERVER	ILLEGAL FUNCTION:
xx	xx	xx	**02**	MB_SERVER	ILLEGAL DATA ADDRESS:
xx	xx	xx	**03**	MB_SERVER	ILLEGAL DATA VALUE:

Unterstützte Funktionscodes und verwendete Parameter:

Funktionscode	**Bit Zugriff**	**16 Bit Zugriff (Register)**	**Gruppe**	**Funktion Beschreibung**
1	x		Coils	Read Coils
2	x		Input Discrete	Read Discrete Inputs
3		x	Holding Register	Read Holding Registers

4		x	Input Register	Read Input Register
5	x		Coils	Write Single Coil
6		x	Holding Register	Write Single Register
15	x		Coils	Write Multiple Coils
16		x	Holding Register	Write Multiple Register
22		x	Holding Register	Mask Write Register
23		x	Holding Register	Read/Write Multiple Register

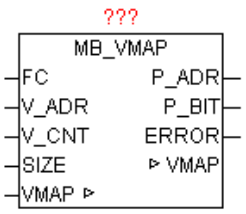
9.15. MB_VMAP

Type Funktionsbaustein:

IN_OUT VMAP : ARRAY [1..10] OF VMAP_DATA (VIRTUAL_MAP Daten)

INPUT FC: INT (Funktionsnummer)
V_ADR: INT (Virtuelle Adressbereich: Startadresse)
V_CNT: INT (Virtuelle Adressbereich: Anzahl der Datenpunkte)
SIZE: INT (Anzahl der MODBUS-Register in Struktur DATA)

OUTPUT: P_ADR: INT (Realer Adressbereich: Startadresse)
P_BIT: INT (Realer Adressbereich: Bitposition)
ERROR: DWORD (Fehlercode)



Der Baustein ermöglicht die Umrechnung von Virtuellen Adressen auf einen Realen Adressbereich im der MODBUS-DATA Struktur. Dazu werden im VMAP-Datenarray Virtuelle Adressbereiche definiert. Wird der Baustein aufgerufen und dabei festgestellt das in den VMAP-Daten nichts eingetragen ist, wird automatisch ein Block angelegt, der den vollen Zugriff auf die ganzen MODBUS-Daten ermöglicht. In jedem Adressblock wird ein auch Watchdog-Timer verwaltet, der bei jedem Zugriff über diesem Block den Timer auf Null setzt. Somit kann einfach durch

vergleichen des TIME_OUT Wertes mit einem Abschaltwert auf Kommunikationsstörungen (keine Aktualisierung) reagiert werden.

Mittels dem Parameter FC wird erkannt um welchen Funktionscode es sich handelt, und ob dabei Register (16Bit) oder einzelne Bits behandelt werden müssen. Die Bitnummer entspricht dem Funktionscode. Das heißt das z.B. BIT5=1 in FC den Funktionscode 5 (Write Single Coil) aktiviert. Durch V_ADR wird die virtuelle Startadresse angegeben (Bei 16Bit Befehlen ist dies eine Register-Adresse und bei Bit Befehlen eine absolute Bitnummer innerhalb eines definierten Blocks. Der Parameter V_CNT definiert die Anzahl der Datenpunkte (Einheit 16 Bit oder Bit je nach Funktionscode). Die Gesamtgröße des MODBUS_ARRAY wird über SIZE (Anzahl WORDS) vorgegeben. Anhand dieser Parameter durchsucht der Baustein die VMAP Datentabelle nach einen passenden Datenblock, und gibt vom ersten korrekten Datenblock die P_ADR als Ergebnis zurück. Der Wert entspricht den realen Index für MODBUS_DATA Array. Bei einem Funktionscode mit Bit Zugriff wird noch zusätzlich die Bit-Position innerhalb P_ADR mit ausgegeben. Ein möglicherweise auftretender Fehler bei der Auswertung wird beim Parameter „Error“ gemeldet (Siehe Error-Tabelle). Der Watchdog-Timer wird bei jeden Zugriff mit einem Funktionscode aus der Gruppe der schreibenden Kommandos zurückgestellt.

Wird keine Sonderbehandlung gewünscht so sind in VMAP keinerlei Einstellungen notwendig, und das MODBUS_ARRAY wird dann 1:1 beim Zugriff abgebildet.

ERROR:

Wert	Beschreibung
0	Keine Fehler
1	Ungültiger Funktionscode
2	Ungültige Datenadresse

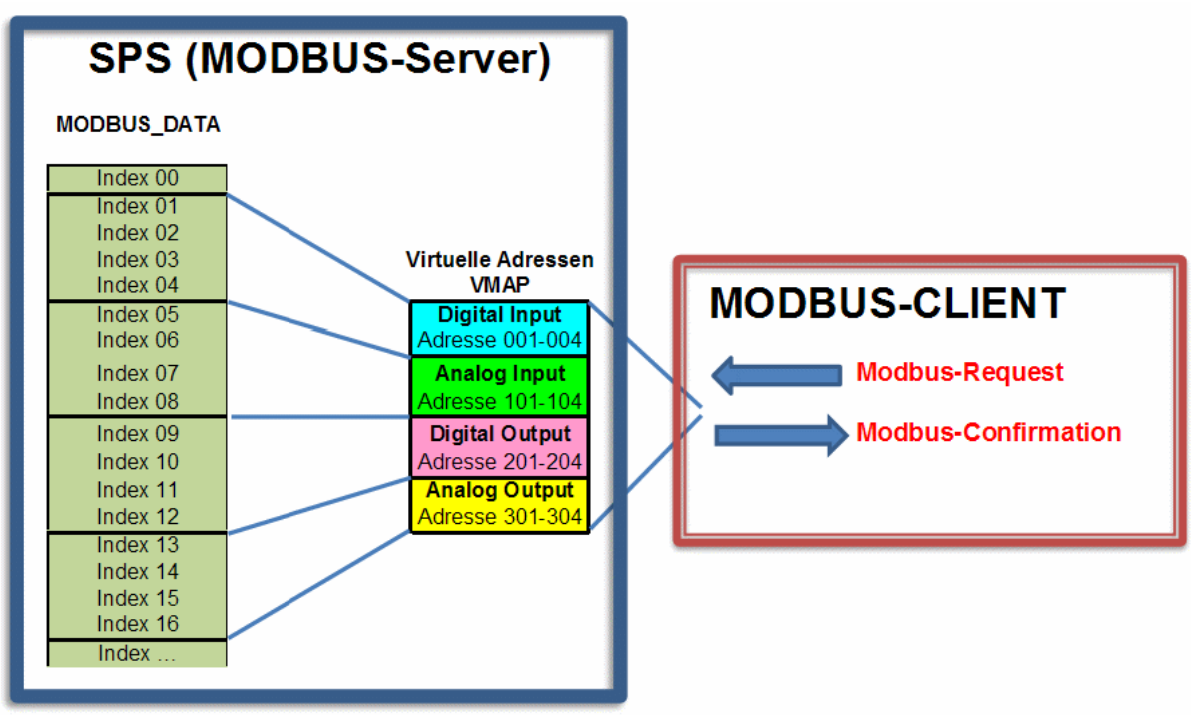
! Hinweis zur Sonderbehandlung des Funktionscode 23 !

Der Modbus-Funktionscode 23 ist ein kombinierter Befehl, weil er aus zwei Aktionen besteht. Zuerst werden Register geschrieben, und danach Register gelesen. Wird festgestellt das Schreib oder Lese-Parameter nicht zulässig sind, so wird keine der beiden Aktionen durchgeführt.

Damit über VMAP auch zwischen dem Lesen und Schreiben unterschieden werden kann, wird der Lesebefehl in VMAP bei FC 23 als BIT23 (Read/Write Multiple Register) , und der Schreibbefehl hingegen über BIT16 (Write Multiple Register) geprüft.

Beispielkonfiguration

```
(* Virtueller Block 1 *)
VMAP[1].FC := DWORD#2#00000000_10000000_00000000_00011100); (FC 2,3,4,23)
VMAP[1].V_ADR := 1;  (* Virtueller Adressbereich: Startadresse *)
VMAP[1].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[1].P_ADR := 1;  (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 2 *)
VMAP[2].FC := DWORD#2#00000000_10000000_00000000_00011000); (FC 3,4,23)
VMAP[2].V_ADR := 101;(* Virtueller Adressbereich: Startadresse *)
VMAP[2].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[2].P_ADR := 5;  (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 3 *)
VMAP[3].FC := DWORD#2#00000000_11000001_10000000_01111010);(FC1,3-6,15-16,23)
VMAP[3].V_ADR := 201;(* Virtueller Adressbereich: Startadresse *)
VMAP[3].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[3].P_ADR := 9;  (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 4 *)
VMAP[4].FC := DWORD#2#00000000_11000001_00000000_01011000); (FC 3,4,6,16,23)
VMAP[4].V_ADR := 301;(* Virtueller Adressbereich: Startadresse *)
VMAP[4].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[4].P_ADR := 12;       (* Realer Adressbereich: Startadresse *)
```



Die Konfiguration ergibt folgende Zugriffs-Matrix:

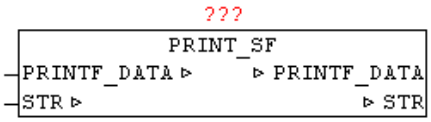
Funktion Beschreibung	Funktionscode	Bit Zugriff	16 Bit Zugriff (Register)	Lesen / Schreiben	Digital Input	Analog Input	Digital Output	Analog Output
Read Coils	**1**	x		Lesen			x	
Read Discrete Inputs	**2**	x		Lesen	x			
Read Holding Registers	**3**		x	Lesen	x	x	x	x
Read Input Register	**4**		x	Lesen	x	x	x	x
Write Single Coil	**5**	x		Schreiben			x	
Write Single Register	**6**		x	Schreiben			x	x
Write Multiple Coils	**15**	x		Schreiben			x	
Write Multiple Register	**16**		x	Schreiben			x	x
Mask Write Register	**22**		x	Schreiben				
Read/Write Multiple Register	**23**		x	Lesen / Schreiben	x	x	x	x

9.16. PRINT_SF

Type Funktionsbaustein:

IN_OUT PRINTF_DATA : ARRAY[1..11] OF STRING(STRING_LENGTH) (Parameterdaten)

STR: STRING(STRING_LENGTH) (Ergebnisstring)



Mit PRINT_SF kann ein STRING mit Teilstrings dynamisch ergänzt werden. Die Position der einzufügenden Teilstrings wird mittels '~' Tilde Zeichen angegeben und die nachfolgenden Nummer definiert die Parameternummer. '~1' bis '~9' werden somit automatisch verarbeitet. Wird beim einfügen des Teilstrings die maximale Anzahl an Zeichen erreicht so wird anstatt des Teilstring nur mit '..' eingefügt.

```
VAR
  LITER : REAL := 545.4;
  FUELLZEIT : INT := 25;
  NAME : STRING := 'Tankinhalt';
  PARA : ARRAY[1..11] OF STRING(STRING_LENGTH);
  PS : PRINT_SF;
END_VAR
PARA[1] := REAL_TO_STRING(Liter); (* Parameter 1: in String wandeln *)
PARA[2] := INT_TO_STRING(Fuellzeit); (* Parameter 2: in String wandeln *)
PARA[3] := NAME; (* Parameter 3: *)
PS.STR := '~3: ~1 Liter, Füllzeit: ~2 Min.'; (* Textausgabe-Maske *)
PS.PRINTF_DATA := PARA; (* Parameter Datenstruktur übergeben *)
PS(); (* Baustein ausführen *)
```

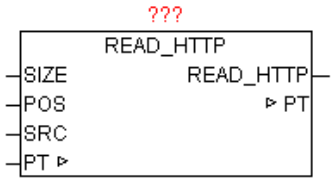
Der String PS.STR hat danach folgenden Inhalt

'Tankinhalt: 545.4 Liter, Füllzeit: 25 Min.'

9.17. READ_HTTP

Type Funktionsbaustein:

INPUT	SIZE : UINT	(Größe des Puffers)
	POS : INT	(Position ab der gesucht wird)
	SRC : STRING	(Suchstring)
IN_OUT	PT : POINTER	(Adresse des Puffers)
OUTPUT	VALUE	(Parameter der Header-Information)



Nach einem erfolgreichem HTTP-GET Request sind immer ein HTTP-Header (Message Header) und ein Nachrichtenkörper (Message Body) im Buffer vorhanden. Im HTTP-Header sind diverse Informationen zur angefragten HTTP-Seite abgelegt. Der nachfolgende Nachrichtenkörper beinhaltet die eigentlichen angefragten Daten. Mittels READ_HTTP können die HTTP-Header-Informationen ausgewertet werden. Der Baustein durchsucht ein beliebiges Array of Byte auf den Inhalt einer Zeichenkette und wertet danach den nachfolgenden Parameter aus, und liefert diesen String als Ergebnis zurück. Die Daten im Buffer werden automatisch auf Großbuchstaben konvertiert, daher müssen auch alle Suchstring bei SRC in Großbuchstaben übergeben werden. Mit POS kann an beliebiger Position mit der Suche begonnen werden. Das erste Element im Array hat die Positionsnummer 1.

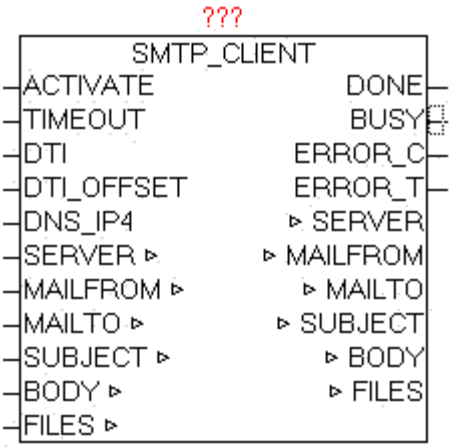
Beispiel einer HTTP-Response (Header-Informationen):

HTTP/1.0 200 OK<CR><LF>
Content-Length: 2165<CR><LF>
Content-Type: text/html<CR><LF>
Date: Mon, 15 Sep 2008 16:59:08 GMT<CR><LF>
Last-Modified: Wed, 18 Jun 2008 12:35:52 GMT<CR><LF>
Mime-Version: 1.0<CR><LF><CR><LF>

Wenn SRC keinen Suchbegriff beinhaltet, wird automatisch die HTTP Version und der HTTP Statuscode im Buffer besucht und ausgewertet. Als Ergebnis wird nach obigen Beispiel „1.0 200 OK“ zurückgegeben. Ist in SRC ein Suchbegriff enthalten wird diese Header-Information im Buffer gesucht und der Wert als String zurückgegeben z.B. 'Content-Length' = „2165“.

9.18. SMTP_CLIENT

Type	Funktionsbaustein:
IN_OUT	SERVER : STRING (URL des SMTP-Server)
	MAILFROM : STRING (Absenderadresse)
	MAILTO : STRING(STRING_LENGTH) (Empfängeradresse)
	SUBJECT : STRING (Betreff-Text)
	BODY : STRING(STRING_LENGTH) (Email-Inhalt)
	FILES : STRING(STRING_LENGTH) (Dateien anhänge)
INPUT	ACTIVATE : BOOL (positive Flanke startet die Abfrage)
	TIMEOUT : TIME (Zeitüberwachung)
	DTI : DT (aktuelles Datum-Uhrzeit)
	DTI_OFFSET : INT (Zeitzonen Offset zu UTC)
	DNS_IP4 : DWORD (IP4-Adresse des DNS-Server)
OUTPUT	DONE : BOOL (Transfer ohne Fehler beendet)
	BUSY : BOOL (Transfer ist aktiv)
	ERROR_C : DWORD (Fehlercode)
	ERROR_T : BYTE (Fehlertype)



Der Baustein SMTP_CLIENT dient zum versenden von klassischen Emails.

Folgenden Funktionen werden unterstützt:
SMTP-Protokoll
Extended-SMTP-Protokoll
Versenden von Betreffzeile, und Inhaltstext

Angabe von Email-Absenderadresse (From:) inklusive „Angezeigter Name“

Angabe von Empfänger(n) (To:)

Angabe von CarbonCopy-Empfänger(n) (Cc:)

Angabe von Blindcopy-Empfänger(n) (Bc:)

Versenden von Datei(en) als Anhang

Authentifizierungs-Verfahren: OHNE,PLAIN,LOGIN,CRAM-MD5

Angabe der PORT-Nummer

Mittels positiver Flanke bei ACTIVATE wird der Übertragungsvorgang gestartet.Der Parameter SERVER enthält den Namen des SMTP-Server und optional den Benutzernamen und das Passwort und eine Port-Nummer. Wird kein Benutzername bzw. Passwort übergeben, so wird nach Standard SMTP vorgegangen.

SERVER: URL-Beispiele:

benutzername:password@smtp_server

benutzername:password@smtp_server:portnummer

smtp_server

Sonderfall:

Befindet sich im Benutzernamen ein '@' so muss dies als '%' Zeichen übergeben werden, und wird danach vom Baustein automatisch wieder korrigiert.

Bei Angabe von Benutzer und Passwort wird Extend-SMTP benutzt, und automatisch das möglichst sicherste Authentifizierungs-Verfahren angewendet. Bei Parameter MAILFROM wird die Absenderadresse angeben:

z.B. oscat@gmx.net

optional kann ein zusätzlicher „Angezeigter Name“ hinzugefügt werden. Dieser wird vom Email-Client automatisch anstatt der echten Absenderadresse angezeigt. Damit kann immer ein leicht erkennbarer Name angewendet werden.

z.B. oscat@gmx.net;Station_01

Der Email-Client zeigt als Absender dann „Station_01“ an. Somit können mehrere Teilnehmer die gleiche Email-Adresse benutzen, jedoch eine eigenen „Alias“ Kennung mitsenden.

Bei Parameter MAILTO können To,Cc,Bc angeben werden. Die verschiedene Empfängergruppen werden mittels '#' als Trennzeichen als Liste

angegeben. Mehrere Adressen innerhalb der selben Gruppe werden mit dem Trennzeichen ';' unterteilt. Es können von jeder Gruppe beliebig viele Empfänger vorgegeben werden, einzige Beschränkung ist die Länge des MAILTO-Strings.

To;To..#Cc;Cc...#Bc;Bc...

Beispiele.

o1@gmx.net;o2@gmx.net#o1@gmx.net#o2@gmx.net

definiert zwei To-Adressen, eine Cc-Adresse und eine Bc-Adresse

##o2@gmx.net

definiert nur eine Bc-Adresse

Mit SUBJECT kann ein Betreff-Text vorgeben werden, sowie bei BODY ein Email Inhalt als Text. Der aktuelle Date/Time Wert muss bei DTI , und bei DTI_OFFSET der Korrekturwert als Offset in Minuten zur UTC (Weltzeit) angegeben werden. Wenn bei DTI die Weltzeit UTC übergeben wird, muss bei DTI_OFFSET 0 übergeben werden.

Es können Dateien als Anhang versendet werden. Die Dateien müssen bei Parameter FILES in Listenform übergeben werden. Es können beliebig viele Dateien vorgegeben werden, einzige Beschränkung ist die Länge des FILES-Strings, und der Speicherplatz des Email-Postfachs (in der Praxis 5-30 Megabyte).

Durch eine zusätzliche optionale Angabe von '#DEL# kann nach erfolgreicher Übertragung der Dateien per Email, das Löschen der Dateien auf der Steuerung ausgelöst werden.

z.B.

FILES: 'log1.csv;log2.csv;#DEL#'

Die beiden Dateien werden nach erfolgreicher Übertragung gelöscht.

Die Überwachungszeit kann bei Parameter TIMEOUT vorgegeben werden. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn bei SERVER ein DNS-Name angegeben wird.Sollten bei der Übertragung Fehler auftreten, werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY = TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE = TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies normalerweise notwendig wäre.

Grundlagen:

http://de.wikipedia.org/wiki/SMTP-Auth

http://de.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

ERROR_T:

Wert	Eigenschaften
1	Störung: DNS_CLIENT Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Störung: SMTP Steuerkanal Die genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVER Die genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUT ERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom SMTP-Server empfangenen Response-Code. Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel: ERROR_T = 5 ERROR_C = 0x0028_00FA Ablauf-Schrittnummer 0x0028 = 40 Response-Code 0x00DC = 250

9.19. SNTP_CLIENT

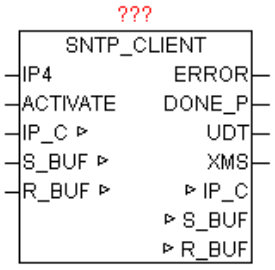
Type Funktionsbaustein:

IN_OUT IP_C : IP_C (Parametrierungsdaten)

S_BUF: NETWORK_BUFFER (Sendedaten)

R_BUF: NETWORK_BUFFER (Empfangsdaten)

INPUT	IP4: DWORD (IP-Adresse des SNTP-Servers)
	ACTIVATE: BOOL (Startet die Abfrage)
OUTPUT	ERROR: DWORD (Fehlercode)
	DONE_P: BOOL (positiver Flanke Fertig ohne Fehler)
	UDT: DT (Datums und Zeit Ausgang als Weltzeit)
	XMS: INT (Millisekunden der Weltzeit UDT)



Der SNTP_CLIENT dient zum synchronisieren der lokalen Uhrzeit mit einem SNTP-Server. Dazu wird das Simple-Network-Time-Protokoll verwendet, das speziell entwickelt wurde , um eine zuverlässige Zeitangabe über Netzwerke mit variabler Paketlaufzeit zu ermöglichen. Das SNTP ist Datentechnisch völlig identisch mit NTP, das heißt hier bestehen keinerlei Unterschiede. Somit können sämtliche gekannte SNTP und NTP Server ob im lokalen Netzwerk als auch über das Internet genutzt werden. Bei IP4 wird die IP-Adresse eines SNTP/NTP Server angegeben. Eine positive Flanke bei ACTIVATE startet die Abfrage. Die vergangene Zeit zwischen Senden und dem Empfang der Zeit wird gemessen und daraus wird eine Zeitkorrektur errechnet. Danach wird die empfangene Zeit wird um diesen Wert korrigiert. Bei erfolgreicher Beendigung gibt DONE_P eine positiven Flanke aus, und die aktuelle Zeit wird bei UDT ausgegeben. Weiters werden auf XMS noch die zugehörigen Sekundenbruchteile als Millisekunden ausgegeben. Die Werte von UDT und XMS sind nur bei DONE_P = TRUE gültig, da es sich um einen statischen Uhrzeitwert handelt, und dienen nur zum Impuls gesteuerten Uhrzeitstellen. ERROR liefert im Fehlerfall die genaue Ursache (Siehe Baustein IP_CONTROL).

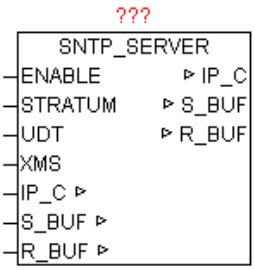
9.20. SNTP_SERVER

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	ENABLE: BOOL (Startet die SNTP-Server)

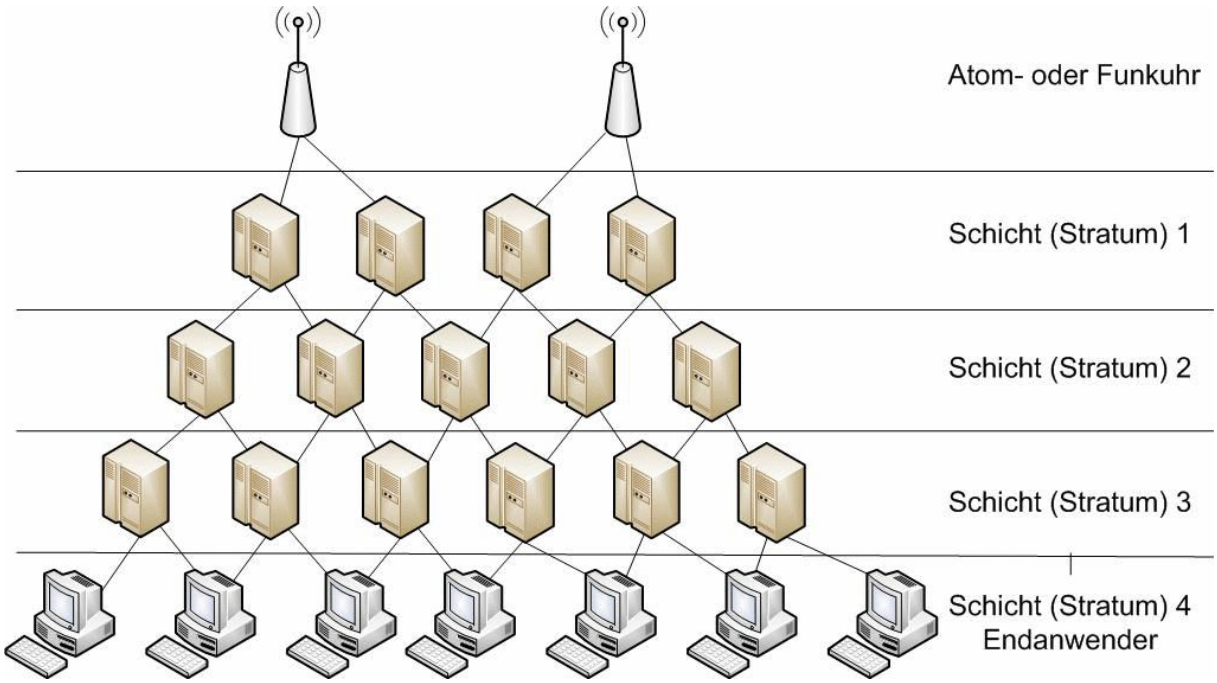
STRATUM: BYTE (Angabe der hierarchische Ebene bzw. Genauigkeit)

UDT: DT (Datums und Zeit Eingang als Weltzeit)

XMS: INT (Millisekunden der Weltzeit UDT)



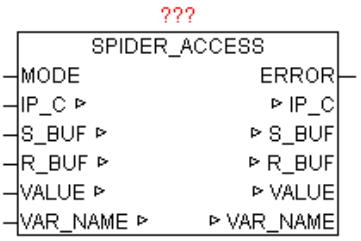
Der Baustein stellt die Funktionalität eines SNTP (NTP) Servers zu Verfügung. Bei ENABLE = TRUE meldet sich der Baustein bei IP_CONTROL an und wartet auf Freigabe der Ressource, sollte diese von anderen Teilnehmern momentan noch belegt sein. Danach wartet der Baustein auf Anfragen von anderen SNTP-Clients und beantwortet diese mit der aktuellen Uhrzeit von UDT und XMS. Solange ENABLE = TRUE ist, wird der Zugriff auf diese Ethernet-Ressource dauerhaft für andere Teilnehmer gesperrt (Exklusiv-Zugriff - wegen Passiv UDP Modus). SNTP nutzt ein hierarchisches System verschiedener Strata. Als Stratum 0 bezeichnet man das exakte Zeitnormal. Die unmittelbar mit ihr gekoppelten Systeme, beispielsweise NTP, GPS oder DCF77 Zeitsignale heißen Stratum 1. Jede weitere abhängige Einheit verursacht einen zusätzlichen Zeitversatz von 10-100ms und erhält bei der Bezeichnung eine höhere Nummer (Stratum 2, Stratum 3 bis 255). Wird kein STRATUM am Baustein angegeben wird STRATUM=1 als Standard verwendet.



Wenn ein SNTP-Client selber eine Uhrzeit mit höheren Stratum als eine SNTP-Server hat, wird die Uhrzeit von diesen mitunter abgelehnt, da diese ungenauer ist ,als die eigene Referenz. Darum ist es wichtig ein logisch richtiges STRATUM vorzugeben. Der Baustein SNTP_CLIENT ignoriert jedoch absichtlich das STRATUM und synchronisiert sich auf jeden Fall mit dem SNTP-Server, da ziemlich jeder SNTP-Server eine genauere Uhrzeit besitzt , als eine SPS.

9.21. SPIDER_ACCESS

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
	VALUE: STRING (Wert der Variable)
	NAME: STRING(40) (Variablename)
INPUT	MODE: BYTE (Betriebsmodus: 1= lesen / 2=schreiben)
ERROR	ERROR: DWORD (Fehlercode)



ERROR:

Wert	Eigenschaften
1	Beim Schreiben von Variablenwerte ist ein Fehler aufgetreten
> 1	Die genaue Bedeutung von ERROR ist beim Baustein HTTP_GET nachzulesen

Mit SPIDER_ACCESS können von Steuerungen die Visualisierungen über Webserver auf Basis „SpiderControl“ von Fa. IniNet Solution GmbH integriert haben, Variablen gelesen und geschrieben werden.

Für folgende Steuerungen gibt es diese Webserver Integration:

Simatic S7 200/300/400
SAIA-Burgess PCD
Wago (750-841)
Beckhoff (CX Reihe)
Phoenix Contact (ILC Reihe)
Selectron
Berthel
Tbox
Beck IPC

Im SPS-Programm der Zielsteuerung müssen die gewünschten Variablen für den Webzugriff freigegeben sein. Da die Kommunikation mittels HTTP (Port 80) durchgeführt wird, kann auch über Firewalls hinweg der Datenaustausch problemlos erfolgen. Es können Globale als auch Instanz-Variablen verarbeitet werden.

Format der Variablen:

Bei einer Globalen Variablen muss nur der normale Variablenname angeben werden. Eine Instanz-Variable muss folgend angegeben werden. „instanzname.Variablenname“

Modus: Lesen

Wird der Parameter MODE auf „1“ gesetzt und bei „NAME“ der Variablenname angegeben, so wird zyklisch eine HTTP Anfrage an den

Webserver (SPS) durchgeführt, und das gemeldete Ergebnis bei „VALUE“ als STRING ausgegeben.

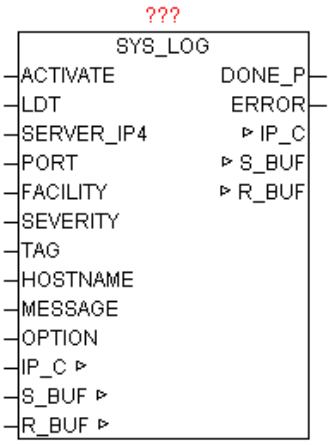
Modus: Schreiben

Wird der Parameter MODE auf „2“ gesetzt und bei „VALUE“ der Variablenwert und bei „NAME“ der Variablenname als STRING angegeben, so wird zyklisch eine HTTP Anfrage an den Webserver (SPS) durchgeführt

Der Modus bzw. der Variablenname kann im zyklischen Betrieb jederzeit geändert werden. Sollten mehrere Variablen verarbeitet werden müssen, so müssen lediglich dementsprechend viele Bausteininstanzen aufgerufen werden.

9.22. SYS_LOG

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Abfrage)
	LDT: DT (Lokalzeit)
	SERVER_IP4: DWORD (IP-Adresse des SYS-LOG Servers)
	PORT: WORD (Port-Nummer des SYS-LOG Servers)
	FACILITY: BYTE (spezifiziert den Dienst oder die Komponente)
	SEVERITY: BYTE (Klassifizierung des Schweregrades)
	TAG: STRING(32) (Prozessname , ID etc.)
	HOSTNAME: STRING (Name oder IP-Adresse des Senders)
	MESSAGE: STRING(STRING_LENGTH) (Nachrichtentext)
	OPTION: BYTE (diverse)
OUTPUT	DONE: BOOL (Abfrage ohne Fehler beendet)
	ERROR: DWORD (Fehlercode)



SYSLOG ist ein Standard zur Übermittlung von Meldungen in einem IP-Rechner-Netzwerk. Das Protokoll ist dabei sehr einfach aufgebaut – der Client sendet eine kurze Textnachricht an den SYSLOG-Empfänger. Der Empfänger wird auch als "syslog daemon" oder "syslog server" bezeichnet. Die Meldungen werden mittels UDP-Port 514 oder über TCP-Port 1468 gesendet und enthalten die Nachricht im Klartext. SYSLOG wird typischerweise für Computersystem-Management und Sicherheits-Überwachung benutzt. Damit ermöglicht es die leichte Integration von verschiedensten LOG-Quellen auf einen zentralen SYSLOG-Server. Die Server-Software gibt es für alle Plattformen mitunter auch als Free / Shareware. Unix bzw. Linux-Systeme haben einen SYSLOG-SERVER schon integriert. Durch eine positive Flanke von ACTIVATE wird aus den Parametern LDT, FACILITY, SEVERITY, TAG, HOSTNAME,MESSAGE eine SYSLOG-Nachricht generiert und an die SERVER_IP4 Adresse gesendet. Mittels OPTION können noch diverse Eigenschaften gesteuert werden (Siehe Tabelle OPTION). Nach erfolgreichen Versenden wird DONE=TRUE, ansonsten wird bei ERROR die konkrete Fehlermeldung ausgegeben (Siehe ERROR von Baustein IP_CONTROL).

Eine Syslog-Message hat folgenden Aufbau

FACILITY,SEVERITY,TIMESTAMP,HOSTNAME,TAG,MESSAGE

Beispiel:

MAIL.ERR: Sep 10 08:31:10 149.100.100.02 PLANT2_PLC1 This is a test message generated by OSCAT SYSLOG

Folgende OPTION können verwendet werden

BIT	Funktion
0	FALSE = mit Facility,Severity-Code TRUE = ohne Facility,Severity-Code
1	FALSE = mit RFC Header

	TRUE = ohne RFC Header (nur die MESSAGE alleine wird versendet)
2	FALSE = mit CR,LF am Ende TRUE = ohne CR,LF am Ende
3	FALSE = UDP Modus TRUE = TCP Modus

Folgende Severity sind als Standard definiert:

Severity	Beschreibung
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Informational
7	Debug

Folgende Facility sind als Standard definiert:

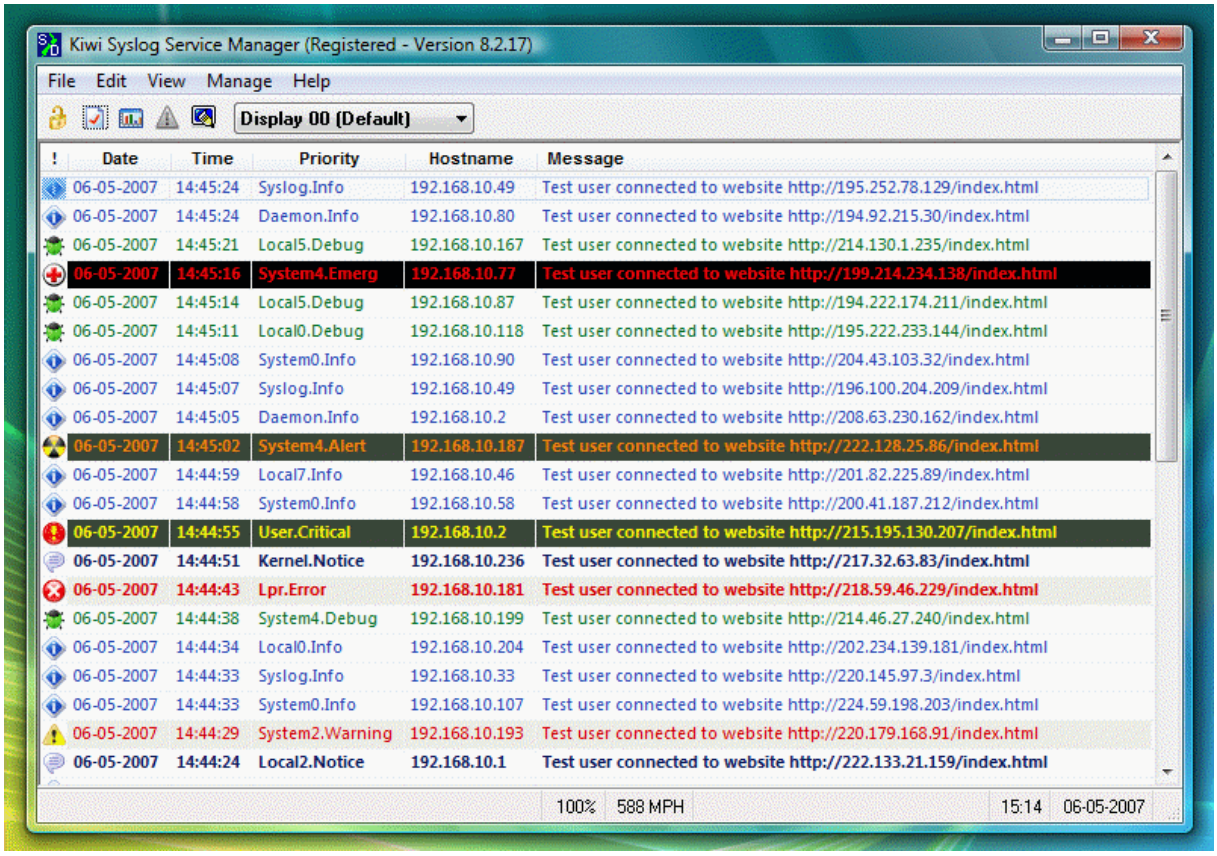
Facility	Beschreibung
00	Kernel message
01	user-level messages
02	mail system
03	system daemons
04	security/authorization messages
05	messages generated internally by syslogd
06	line printer subsystem
07	network news subsystem
08	UUCP subsystem
09	clock daemon
10	security/authorization messages

11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local10
17	local11
18	local12
19	local13
20	local14
21	local15
22	local16
23	local17

Für allgemeine syslog-Nachrichten sind die Facility-Werte 16-23 vorgesehen (local0 bis local7). Es ist aber durchaus zulässig, auch die vordefinierten Werte 0 bis 15 für eigene Zwecke zu verwenden.

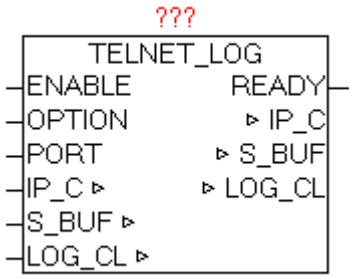
Mit Hilfe von Facility und Severity kann später auf den SYSLOG-Server (Datenbank) sehr leicht nach bestimmten Meldungen gefiltert werden, wie beispielsweise: "Erfasse alle Mailserver-Nachrichten vom Schweregrad error".

Beispiel (Screenshot) eines SYSLOG-Servers für Windows



9.23. TELNET_LOG

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	LOG_CL: LOG_CONTROL (LOG-Daten)
INPUT	S_BUF_SIZE: UINT (Größe des S_BUF)
	ENABLE: BOOL (TELNET Server freigeben)
	OPTION: BYTE (Sende-Optionen)
	PORT: WORD (Port Nummer)
OUTPUT	READY: BOOL (TELNET Client hat Verbindung aufgebaut)



TELNET_LOG wird benutzt um alle Meldungen die sich im LOG_CONTROL Ring-buffer befinden über TELNET auszugeben. Durch „ENABLE“ kann der Baustein aktiviert werden. Bei Parameter PORT kann die gewünschte Port-Nummer vorgegeben werden, wird diese Parameter nicht belegt so wird standardmäßig Port 23 verwendet.

Mittels Parameter OPTION können verschiedene Optionen gewählt werden (Siehe Tabelle OPTION). Wird der Parameter OPTION nicht beschaltet, so wird folgende Einstellung standardmäßig verwendet.

OPTION = BYTE#2#1000_1100;

Sobald sich ein ein TELNET-Client mit dem Baustein verbindet, wird dies über Parameter „READY“ angezeigt. Daraufhin werden automatisch alle Meldungen an TELNET ausgegeben. Sobald im weiteren Verlauf neue Meldungen in LOG_CONTROL auflaufen werden diese immer wieder automatisch auch ausgegeben. Bei einen erneuten Verbindung ab/aufbau werden wieder alles Meldungen erneut ausgegeben. Die meisten TELNET-Clients bieten auch die Möglichkeit den Datenstrom in eine Datei umzuleiten, um hier eine langfristige Datenarchivierung zu ermöglichen.

OPTION:

BIT	Funktion	Beschreibung
0	SCREEN_INIT	Nach dem Verbindungsaufbau mit der TELNET-Konsole wird der gesamte Bildschirm gelöscht. Wenn die OPTION COLOR aktiviert ist, wird der Bildschirm mit BACK_COLOR gelöscht.
1	AUTOWRAP	Bei AUTOWRAP=1 wird der Schreib-Cursor bei erreichen des Zeilenendes automatisch auf eine nächste Zeile gesetzt. Wenn bei der Textausgabe die X,Y Positionen immer mit angegeben werden , ist es besser wenn AUTOWRAP=0 ist.
2	COLOR	Aktiviert die den Farbe-Modus , dabei werden BACK_COLOR und FRONT_COLOR bei der Ausgabe angewandt.
3	NEW_LINE	Bei NEW_LINE=1 wird automatisch am Ende des Textes ein Carriage Return und Line-Feed angehängt. So dass die nächste Textausgabe in einer neuen Zeile beginnt. Dies ist aber nur dann Sinnvoll, wenn keine X_POS und Y_POS vor-

		gegeben werden.
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Verhindert das die Daten im Buffer sofort gesendet werden. Nur wenn der Buffer komplett gefüllt ist, oder diese Option deaktiviert ist, werden die Daten versendet. Ermöglicht das schnelle Senden von vielen Texten im selben Zyklus

9.24. TELNET_PRINT

Type Funktionsbaustein:

IN_OUT IP_C : IP_C (Parametrierungsdaten)

S_BUF: NETWORK_BUFFER (Sendedaten)

INPUT TEXT: STRING(STRING_LENGTH) (Ausgabe Text)

S_BUF_SIZE: UINT ((Größe des S_BUF-Puffers)

ENABLE: BOOL (Freigabe Kommunikation)

SEND: BOOL (positive Flanke - Sendeanstoß)

OPTION: BYTE (Sende-Optionen)

BACK_COLOR: BYTE (Hintergrundfarbe)

FRONT_COLOR: BYTE (Vordergrundfarbe)

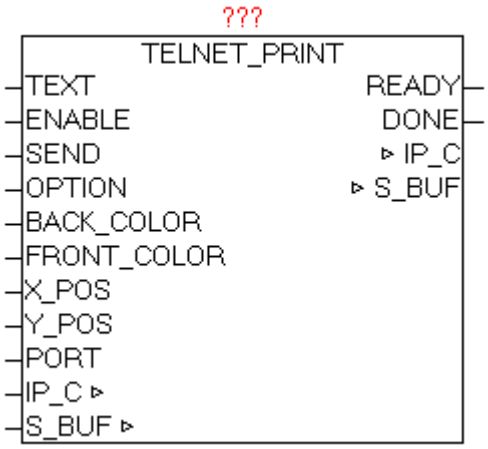
X_POS: BYTE (X-Koordinate der Schreibposition)

Y_POS: BYTE (Y-Koordinate der Schreibposition)

PORT: WORD (Port-Nummer)

OUTPUT: READY: BOOL (Baustein bereit)

DONE: BOOL (positive Flanke - Senden beendet)



Der Baustein ermöglicht die einfache Ausgabe von Texten an eine TELNET-Konsole. Beim Parameter TEXT wird der gewünschte String übergeben. Um den Baustein für die Kommunikation freizuschalten muss ENABLE=1 gesetzt werden, damit erfolgt die Anmeldung beim IP_CONTROL. Bei Parameter PORT wird die gewünschte Port-Nummer vorgegeben, wird der Parameter nicht beschalten , wo wird der Standard-Telnet-Port 23 verwendet. Mittels BACK_COLOR und FRONT_COLOR können die gewünschten Farben vorgegeben werden, vorausgesetzt die Funktion ist Parameter OPTION aktiviert. Die Parameter X_POS und Y_POS geben die gewünschte Koordinate der TEXT Ausgabe an. Wird bei X_POS und Y_POS der WERT „0“ angegeben, so ist die Textpositionierung inaktiv, und die Texte werden immer an der aktuellen Schreib-Cursor Position angehängt. Die Standard Telnet-Konsole erlaubt eine X_POS (Horizontal) von 1 bis 80 und eine Y_POS (Vertikal) von 1 bis 25. Das Verhalten hier kann wiederum mittels OPTION beeinflusst werden (Autowrap, Carriage-Return, Line-Feed, Buf_Flush etc..). Wenn eine große Menge an Texten auf einmal ausgegeben werden muss, so kann eine Bufferung aktiviert werden, sodass die Daten erst geschrieben werden wenn entweder der Buffer voll ist (dies wird vom Baustein selbstständig veranlasst), oder dies durch den geänderten OPTION Parameter signalisiert wird. Durch SEND=1 werden die Daten in den Buffer geschrieben. Die Parameter dürfen erst wieder verändert werden wenn READY=1 ist, und mittels DONE wird die erfolgte Datenübernahme als positive Flanke angezeigt.

OPTION:

BIT	Funktion	Beschreibung
0	SCREEN_INIT	Nach dem Verbindungsaufbau mit der TELNET-Konsole wird der gesamte Bildschirm gelöscht. Wenn die OPTION COLOR aktiviert ist, wird der Bildschirm mit BACK_COLOR gelöscht.
1	AUTOWRAP	Bei AUTOWRAP=1 wird der Schreib-Cursor bei erreichen des Zeilenendes automatisch auf eine nächste Zeile gesetzt. Wenn bei der Textausgabe die X,Y Positionen immer

		mit angegeben werden , ist es besser wenn AUTOWRAP=0 ist.
2	COLOR	Aktiviert die den Farbe-Modus , dabei werden BACK_COLOR und FRONT_COLOR bei der Ausgabe angewandt.
3	NEW_LINE	Bei NEW_LINE=1 wird automatisch am Ende des Textes ein Carriage Return und Line-Feed angehängt. So dass die nächste Textausgabe in einer neuen Zeile beginnt. Dies ist aber nur dann Sinnvoll, wenn keine X_POS und Y_POS vorgegeben werden.
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Verhindert das die Daten im Buffer sofort gesendet werden. Nur wenn der Buffer komplett gefüllt ist, oder diese Option deaktiviert ist, werden die Daten versendet. Ermöglicht das schnelle Senden von vielen Texten im selben Zyklus

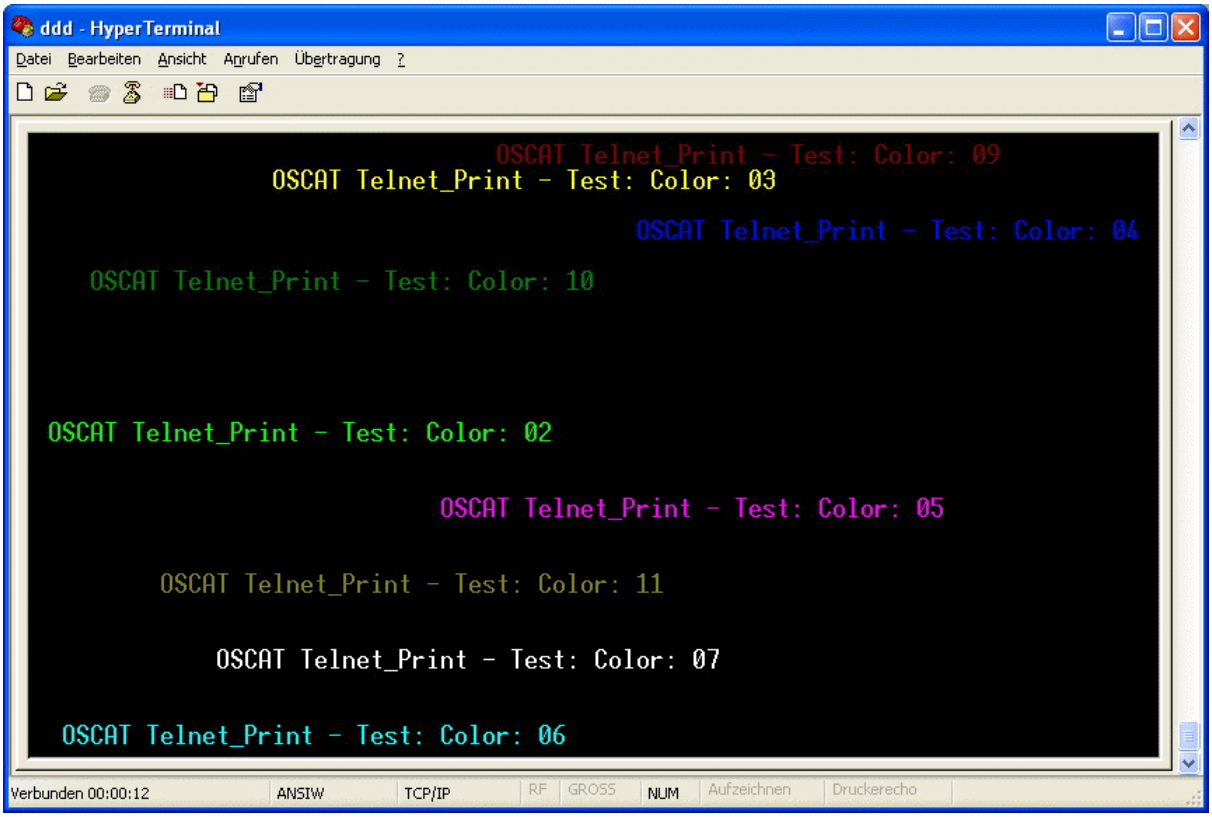
FRONT_COLOR:

Byte	**Farbe**	**Byte**	**Farbe**
0	Black	16	Flashing Black
1	Light Red	17	Flashing Light Red
2	Light Green	18	Flashing Light Green
3	Yellow	19	Flashing Yellow
4	Light Blue	20	Flashing Light Blue
5	Pink / Light Magenta	21	Flashing Pink / Light Magenta
6	Light Cyan	22	Flashing Light Cyan
7	White	23	Flashing White
8	Black	24	Flashing Black
9	Red	25	Flashing Red
10	Green	26	Flashing Green
11	Brown	27	Flashing Brown
12	Blue	28	Flashing Blue
13	Purple / Magenta	29	Purple / Magenta

14	Cyan	30	Flashing Cyan
15	Gray	31	Flashing Gray

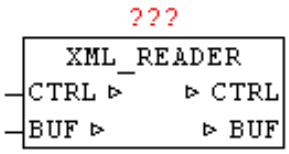
BACK_COLOR:

Byte	Farbe
0	Black
1	Red
2	Green
3	Brown
4	Blue
5	Purple / Magenta
6	Cyan
7	Gray



9.25. XML_READER

Type	Funktionsbaustein:
IN_OUT	CTRL : XML_CONTROL
	(Steuer und Statusdaten)
	BUF: NETWORK_BUFFER (Empfangsdaten)



Mittels XML_READER ist es möglich so genannte 'Well-formed' XML-Dokumente zu parsen. Hierbei werden nicht wie bei Hochsprachen üblich, die ganze XML-Daten eingelesen und als Datenstruktur im Speicher abgelegt, sonder es wird eine sehr Ressourcen schonende Variante angewandt. Der XML_READER liest XML-Daten als sequentiellen Datenstrom aus dem Buffer und meldet die im COMMAND definierte Elemente-Typen automatisch zurück.

Bei XML wird strikt zwischen Groß- und Kleinschreibung unterschieden. Ein XML-gerechtes Dokument besteht aus Elementen, Attributen, ihren Wertzuweisungen, und dem Inhalt der Elemente, der aus Text oder aus untergeordneten Elementen bestehen kann, die ihrerseits wieder Attribute mit Wertzuweisungen und Inhalt haben können. Es gibt Elemente mit und ohne Attribute, Elemente, innerhalb deren viele andere Elemente vorkommen können, und solche, innerhalb deren nur Text vorkommen kann, und sogar leere Elemente, die keinen Inhalt haben können. Die Struktur, die aus diesen Bestandteilen und ihren Grundregeln entsteht, lässt sich als Baumstruktur begreifen. Elemente bestehen immer aus Tags und End-Tags. Attribute sind zusätzliche Informationen zu Elementen. Es sind auch Kommentar-Elemente erlaubt , jedoch dürfen sich diese beim XML_READER nicht zwischen Start- und End-Tags von Elementen befinden. Die möglichen DTD – Document Type Definition werden nur als DTD-Element gemeldet, jedoch nicht weiter vom XML_READER inhaltlich ausgewertet und angewandt. Mit einem CDATA-Abschnitt wird einem Parser mitgeteilt, dass kein Markup folgt, sondern normaler Text, der mittels Start-End-Block zurück gemeldet.

Vor dem ersten Aufruf des XML_READER müssen ein paar Parameter in der CTRL Datenstruktur initialisiert werden. Mittels CTRL.START_POS und CTRL.STOP_POS wird der Beginn und das Ende der XML-Daten im Buffer

definiert. Mit CTRL.COMMAND kann einerseits ein Initialisierung (BIT15=TRUE) angestoßen werden, und mit Bit 0-14 kann definiert werden welche Element/Datentypen zurückgemeldet werden sollen. Dabei entsprechen die Typ-Codes der nachfolgenden Tabelle genau der Bit-Nummer die im CTRL.COMMAND dann jeweils auf TRUE gesetzt werden müssen.

Es wird immer versucht den Text von Element, Attribute, Value und Path in gesamter Länge an zugehörigen STRINGS zu übergeben. Bei STRINGS größer 255 Zeichen werden diese linksbündig abgeschnitten, jedoch mittels Block-Start und Block-End Parameter sehr wohl zurückgemeldet, so dass diese nachträglich trotzdem komplett ausgewertet werden können. Die BLOCK-START/STOP Index werden aber immer parallel zu den STRINGS mit übergeben. Wird der PATH-STRING größer 255 Zeichen so wird die PATH-Verfolgung deaktiviert, sodass nur noch „OVERFLOW“ als Text eingetragen ist.

Da bei sehr großen und komplexen XML-Daten nicht klar ist, wie lange es dauert bis der Baustein Daten zum Rückmelden findet, ist eine WATCH-DOG Funktion integriert. Hiermit kann eine maximale Bearbeitungszeit parametriert werden. Beim Überschreiten der Zeit wird der Baustein-Aufruf automatisch abgebrochen, und im nächsten Zyklus wieder an gleicher Stelle fortgesetzt Dabei wird der Typ-Code 98 zurückgemeldet.

Folgende Typ-Kennungen sind definiert.

Typ (Code)	**Datentyp**	Beschreibung
00	unbekannt	Undefiniertes Element gefunden
01	TAG (Element)	Beginn – des Element Datenzeiger für Element über BLOCK1
02	END-TAG (Element)	Ende – des Element
03	TEXT	Inhalt eines Element Datenzeiger für Value über BLOCK1
04	ATTRIBUTE	Attribute eines Element Datenzeiger für Attribute über BLOCK1 Datenzeiger für Value über BLOCK2
05	TAG (Processing Instruction)	Anweisungen für die Verarbeitung Datenzeiger für Element über BLOCK1
12	CDATA	Inhaltlich nicht analysierter TEXT Datenzeiger für Value über BLOCK1
13	COMMENT	Kommentarzeilen

		Datenzeiger für Value über BLOCK1
14	DTD	Dokumenten Typ Deklaration Datenzeiger für Value über BLOCK1
98	WATCHDOG	Maximale Durchlaufzeit erreicht- Abbruch
99	ENDE	Keine weiteren Elemente vorhanden

XML-Beispiel

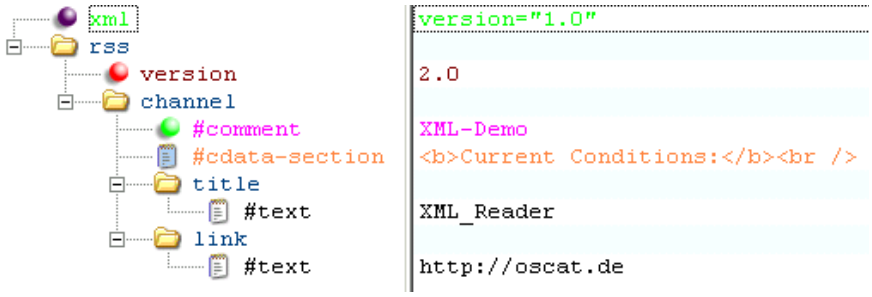
Flache Darstellung

<?xml version="1.0" ?><rss version="2.0"><channel><!-- XML-Demo --><![CDATA[CurrentConditions:
]]><title>XML_Reader</title>
<link>http://oscat.de</link></channel></rss>

Darstellung der Ebenen (ohne Processing-Instruction)

```
-<rss version="2.0">
  -<channel>
     <!-- XML-Demo  -->
     <b>Current Conditions:</b><br />
     <title>XML_Reader</title>
     <link>http://oscat.de</link>
   </channel>
 </rss>
```

Darstellung als Baumansicht mit Elementtypen



Legende:

Anwendungs-Beispiel:

```
  CASE STATE OF
00:
  STATE := 10;
  CTRL.START_POS := HTTP_GET.BODY_START; (Index des ersten Zeichen *)
  CTRL.STOP_POS  := HTTP_GET.BODY_STOP;  (Index des letzten Zeichen *)
  CTRL.COMMAND   := WORD#2#11111111_11111111; (* Init + alle Elemente melden *)
10:
  (* XML Daten seriell lesen *)
  XML_READER.CTRL := CTRL;
  XML_READER.BUF := BUFFER;
  XML_READER();
  CTRL := XML_READER.CTRL;
  BUFFER := XML_READER.BUF;

  IF CTRL.TYP = 99 THEN
    STATE := 20; (* Exit - keine weiteren Elemente vorhanden *)
  ELSIF CTRL.TYP < 98 THEN (* bei Timeout (Code 98) nichts machen *)
    (* Auswertung der XML-Elemente über CTRL-Datenstruktur *)
  END_IF;
20:
   (* sonstiges..... *)
END_CASE;
```

Folgende Information werden dabei über die CTRL-Datenstruktur ausgegeben

```
-------- erster Durchlauf --------
COUNT:          1
TYPE:           5          (OPEN ELEMENT - PROCESSING
INSTRUCTION)
LEVEL:          1
ELEMENT:        'xml'
PATH:           '/xml'
-------- nächster Durchlauf --------
COUNT:          2
TYPE:           4          (ATTRIBUTE)
LEVEL:          1
ELEMENT:        'xml'
ATTRIBUTE:                 'version'
VALUE:          '1.0'
PATH:           '/xml'
```

```
-------- nächster Durchlauf --------
COUNT:          3
TYPE:           2       (CLOSE ELEMENT)
LEVEL:          0
ELEMENT:        'xml'
PATH:           ''
-------- nächster Durchlauf --------
COUNT:          4
TYPE:           1       (OPEN ELEMENT - Standard)
LEVEL:          1
ELEMENT:        'rss'
PATH:           '/rss'
-------- nächster Durchlauf --------
COUNT:          5
TYPE:           4       (ATTRIBUTE)
LEVEL:          1
ELEMENT:        'rss'
ATTRIBUTE:              'version'
VALUE:          '2.0'
PATH:           '/rss'
-------- nächster Durchlauf --------
COUNT:          6
TYPE:           1       (OPEN ELEMENT - Standard)
LEVEL:          2
ELEMENT:        'channel'
PATH:           '/rss/channel'
-------- nächster Durchlauf --------
COUNT:          7
TYPE:           13      (COMMENT-ELEMENT)
LEVEL:          2
VALUE:          ' XML-Demo '
PATH:           '/rss/channel'
-------- nächster Durchlauf --------
COUNT:          8
TYPE:           12      (CDATA)
LEVEL:          2
VALUE:          '<b>Current Conditions:</b><br />'
PATH:           '/rss/channel'
-------- nächster Durchlauf --------
COUNT:          9
TYPE:           1       (OPEN ELEMENT - Standard)
LEVEL:          3
ELEMENT:        'title'
PATH:           '/rss/channel/title'
-------- nächster Durchlauf --------
COUNT:          10
TYPE:           3       (TEXT)
LEVEL:          3
```

```
ELEMENT:        'title'
VALUE:          ' XML_Reader'
PATH:           '/rss/channel/title'
-------- nächster Durchlauf --------
COUNT:          11
TYPE:           2       (CLOSE ELEMENT)
LEVEL:          2
ELEMENT:        'title'
PATH:           '/rss/channel'
-------- nächster Durchlauf --------
COUNT:          12
TYPE:           1       (OPEN ELEMENT - Standard)
LEVEL:          3
ELEMENT:        'link'
PATH:           '/rss/channel/link'
-------- nächster Durchlauf --------
COUNT:          13
TYPE:           3       (TEXT)
LEVEL:          3
ELEMENT:        'link'
VALUE:          'http://oscat.de'
PATH:           '/rss/channel/link'
-------- nächster Durchlauf --------
COUNT:          14
TYPE:           2       (CLOSE ELEMENT)
LEVEL:          2
ELEMENT:        'link'
PATH:           '/rss/channel'
-------- nächster Durchlauf --------
COUNT:          15
TYPE:           2       (CLOSE ELEMENT)
LEVEL:          1
ELEMENT:        'channel'
PATH:           '/rss'
-------- nächster Durchlauf --------
COUNT:          16
TYPE:           2       (CLOSE ELEMENT)
LEVEL:          0
ELEMENT:        'rss'
PATH:           ''
-------- nächster Durchlauf --------
COUNT:          17
TYPE:           99      (EXIT – END OF DATA)
```

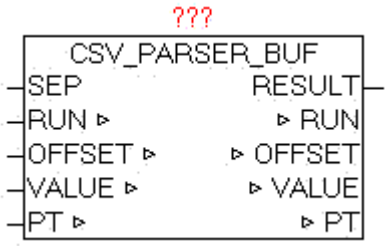
10. File-System

10.1. CSV_PARSER_BUF

Type Funktionsbaustein

IN_OUT SEP : BYTE (Trennzeichen)
RUN : BYTE (Befehlscode für aktuelle Aktion)
OFFSET : UDINT (aktueller Datei-Offset der Abfrage)
VALUE : STRING(STRING_LENGTH) (Wert eines Schlüssels)
PT: NETWORK_BUFFER (Lese Datenbuffer)

OUTPUT: RESULT : BYTE (Ergebnis der Abfrage)



Der Baustein CSV_PARSER_BUF ermöglicht die Auswertung der Elemente die im Buffer enthalten sind. Die Anzahl der enthaltenen Daten wird über PT.SIZE angegeben. Das Trennzeichen wird bei Parameter „SEP“ angegeben. Die Suche nach Elementen beginnt immer abhängig vom übergebenen „OFFSET“, somit ist es sehr leicht möglich an bestimmten Stellen zu suchen, um nicht immer den gesamten Buffer durchsuchen zu müssen. Zu Beginn sollte Standardmäßig mit OFFSET 0 begonnen werden (muss aber nicht !).

Bei der Abfrage von Elementen des Buffers gibt es verschiedene Vorgangsweisen. Dies ist natürlich abhängig vom Inhalt bzw. von der Struktur der Daten.

Elemente auswerten:

Wird bei SEP 0 angeben, so werden immer ganze Zeilen ausgewertet und bei Parameter „VALUE“ ausgegeben. Sind die Elemente im Buffer wie CSV-

Dateien (Excel) strukturiert, so kann bei SEP als Trennzeichen ',' oder ähnliches angeben werden. Mittels RUN = 1 wird die Auswertung gestartet. Da nicht absehbar ist wie lange die suche dauert, ist eine WatchDog Funktion integriert, die die Suche für den aktuellen Zyklus unterbricht, dabei wird bei RESULT = 5 ausgegeben und RUN bleibt unverändert. Im nächsten Zyklus wird die Auswertung automatisch fortgesetzt. Sobald das nächste Element erkannt wurde, wird das Element bei VALUE ausgegeben, und RESULT wird 1 . Ist das Element jedoch gleichzeitig das letzte in einer Zeile dann wird RESULT = 2 ausgegeben. Sobald das Ende der Daten erreicht wurde wird bei RESULT = 10 ausgegeben. Immer erst wenn RUN = 0 ausgegeben wird , definiert RESULT das Ergebnis. Ist ein Element länger als die vorgegeben maximale Länge (STRING_LENGTH) so werden die Zeichen automatisch abgeschnitten. Der Parameter OFFSET wird vom Baustein automatisch nach jedem Ergebnis ausgegeben, kann aber vor jeder neuen Auswertung gezielt vorgegeben werden.

Beispiel 1

Daten zeilenweise auswerten:

Zeile 1<CR,LF>
Zeile 2<CR,LF>

Vorgabe: Offset 0 , SEP = 0 und RUN = 1
VALUE = 'Zeile 1', RUN = 0 , RESULT = 2
Vorgabe: RUN = 1
VALUE = 'Zeile 2', RUN = 0 , RESULT = 2
RUN wieder auf 1 setzen
VALUE = '', RUN = 0 , RESULT = 10

Beispiel 2

Daten als einzelne Elemente auswerten:

10,20<CR,LF>
a,b<CR,LF>

Offset 0 , SEP = ',' und RUN = 1
VALUE = '10', RUN = 0 , RESULT = 1
RUN wieder auf 1 setzen
VALUE = '20', RUN = 0 , RESULT = 2
RUN wieder auf 1 setzen
VALUE = 'a', RUN = 0 , RESULT = 1
RUN wieder auf 1 setzen
VALUE = 'b', RUN = 0 , RESULT = 2

RUN wieder auf 1 setzen
VALUE = '', RUN = 0 , RESULT = 10

RUN : Funktionsübersicht

RUN	Funktion
0	Keine Funktion durchführen – bzw. letzte Funktion ist beendet
1	Element auswerten

RESULT : Ergebnis - Rückmeldung

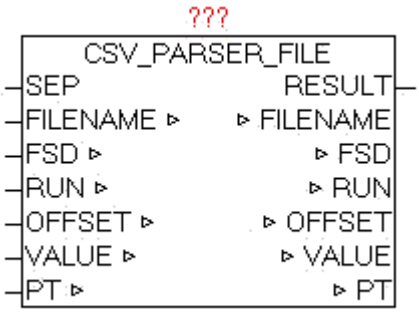
RESULT	Beschreibung
1	Element gefunden
2	Element und Zeilenende erkannt
5	Aktuelle Abfrage läuft noch – Baustein weiter zyklisch aufrufen !
10	Nichts gefunden – Daten-Ende erreicht

10.2. CSV_PARSER_FILE

Type Funktionsbaustein

IN_OUT SEP : BYTE (Trennzeichen)
FILENAME : STRING (Dateiname)
FSD : FILE_SERVER_DATA (Datei Schnittstelle)
RUN : BYTE (Befehlscode für aktuelle Aktion)
OFFSET : UDINT (aktueller Datei-Offset der Abfrage)
VALUE : STRING(STRING_LENGTH) (Wert eines Schlüssels)
PT: NETWORK_BUFFER (Lese Datenbuffer)

OUTPUT: RESULT : BYTE (Ergebnis der Abfrage)



Der Baustein CSV_PARSER_FILE ermöglicht die Auswertung der Elemente einer beliebig großen Datei die zur Verarbeitung automatisch Blockweise in den Lese Datenbuffer eingelesen wird. Das Trennzeichen wird bei Parameter „SEP“ angegeben. Der Name der Datei wird bei Parameter „FILENAME“ übergeben. Die Suche nach Elementen beginnt immer abhängig vom übergebenen „OFFSET“, somit ist es sehr leicht möglich an bestimmten Stellen zu suchen, um nicht immer die gesamte Datei durchsuchen zu müssen. Zu Beginn sollte Standardmäßig mit OFFSET 0 begonnen werden (muss aber nicht !).

Bei der Abfrage von Elementen der Datei gibt es verschiedene Vorgangsweisen. Dies ist natürlich abhängig vom Inhalt bzw. von der Struktur der Daten.

Elemente auswerten:

Wird bei SEP 0 angeben, so werden immer ganze Zeilen ausgewertet und bei Parameter „VALUE“ ausgegeben. Sind die Elemente in der Datei wie CSV-Dateien (Excel) strukturiert, so kann bei SEP als Trennzeichen ',' oder ähnliches angeben werden. Mittels RUN = 1 wird die Auswertung gestartet. Da nicht absehbar ist wie lange die suche dauert, ist eine WatchDog Funktion integriert, die die Suche für den aktuellen Zyklus unterbricht, dabei wird bei RESULT = 5 ausgegeben und RUN bleibt unverändert. Im nächsten Zyklus wird die Auswertung automatisch fortgesetzt. Sobald das nächste Element erkannt wurde, wird das Element bei VALUE ausgegeben, und RESULT wird 1 . Ist das Element jedoch gleichzeitig das letzte in einer Zeile dann wird RESULT = 2 ausgegeben. Sobald das Ende der Daten erreicht wurde wird bei RESULT = 10 ausgegeben. Immer erst wenn RUN = 0 ausgegeben wird , definiert RESULT das Ergebnis. Ist ein Element länger als die vorgegeben maximale Länge (STRING_LENGTH) so werden die Zeichen automatisch abgeschnitten. Der Parameter OFFSET wird vom Baustein automatisch nach jedem Ergebnis ausgegeben, kann aber vor jeder neuen Auswertung gezielt vorgegeben werden.

Beispiel 1

Text-Datei zeilenweise auswerten:

Zeile 1<CR,LF>
Zeile 2<CR,LF>

Vorgabe: Offset 0 , SEP = 0 und RUN = 1
VALUE = 'Zeile 1', RUN = 0 , RESULT = 2
Vorgabe: RUN = 1
VALUE = 'Zeile 2', RUN = 0 , RESULT = 2
RUN wieder auf 1 setzen
VALUE = '', RUN = 0 , RESULT = 10

Beispiel 2

Text-Datei als einzelne Elemente auswerten:

10,20<CR,LF>
a,b<CR,LF>

Offset 0 , SEP = ',' und RUN = 1
VALUE = '10', RUN = 0 , RESULT = 1
RUN wieder auf 1 setzen
VALUE = '20', RUN = 0 , RESULT = 2
RUN wieder auf 1 setzen
VALUE = 'a', RUN = 0 , RESULT = 1
RUN wieder auf 1 setzen
VALUE = 'b', RUN = 0 , RESULT = 2
RUN wieder auf 1 setzen
VALUE = '', RUN = 0 , RESULT = 10

Wenn der Datei-Zugriff nicht mehr benötigt wird, muss vom Anwender entweder durch Nutzung von AUTO_CLOSE oder durch MODE 5 (Datei schließen) über den FILE_SERVER die Datei wieder geschlossen werden.

RUN : Funktionsübersicht

RUN	Funktion
0	Keine Funktion durchführen – bzw. letzte Funktion ist beendet
1	Element auswerten

RESULT : Ergebnis - Rückmeldung

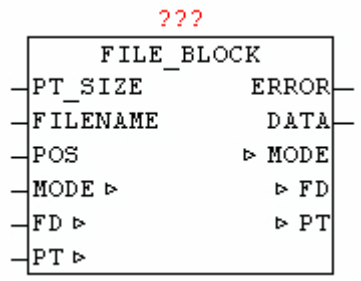
RESULT	Beschreibung
1	Element gefunden

2	Element und Zeilenende erkannt
5	Aktuelle Abfrage läuft noch – Baustein weiter zyklisch aufrufen !
10	Nichts gefunden – Daten-Ende erreicht

10.3. FILE_BLOCK

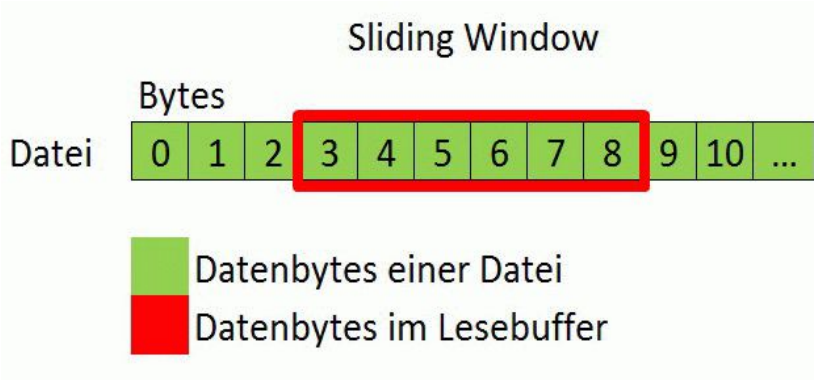
Type Funktionsbaustein

INPUT PT_SIZE : UINT (Anzahl der Bytes im Buffer)
FILENAME : STRING (Dateiname)
POS : UDINT (aktuelle Datei Leseposition)
OUTPUT: ERROR : BYTE (Fehlercode – Siehe Baustein FILE_SERVER)
DATA: BYTE (BYTE der angeforderte Dateiposition)
IN_OUT MODE : BYTE (Aktueller Betriebsmodus)
FD : FILE_SERVER_DATA (File Schnittstelle)
PT: NETWORK_BUFFER (Lesedaten)



Der Baustein FILE_BLOCK ermöglicht den Zugriff auf beliebig große Dateien indem immer ein Datenblock in einem Lesepuffer bereitgehalten wird. Wenn das angeforderte Byte einer Datei sich nicht im zuletzt eingelesenen Datenblock befindet, wird automatisch ein neuer passender Datenblock eingelesen, und das gewünschte Byte ausgegeben. Je größer der Lesepuffer ist, umso seltener muss wieder ein Block neu gelesen

werden. Optimal ist dabei ein linearer Zugriff auf die Bytes, sodass möglichst selten ein Datenblock neu gelesen werden muss.



Ablauf:

Bei Parameter FILENAME wird der Name der zu lesenden Datei angegeben, und mittels PT_SIZE wird die Größe des Lesepuffer in Bytes angegeben. Der Wert bei Parameter POS gibt die genaue Datenposition innerhalb der Datei an, das ausgelesenen werden soll. Der Vorgang wird durch ein setzen von MODE auf 1 ausgelöst. Danach wird automatisch überprüft ob sich das gewünschte Datenbyte schon im Lesepuffer befindet. Wenn nicht dann wird ein neuer passender Datenblock in den Lesepuffer kopiert, und das gewünschte Datenbyte am Parameter DATA ausgegeben. Solange dieser Vorgang läuft und noch nicht beendet ist bleibt MODE auf 1, und wird erst nach Beendigung des Vorgangs vom Baustein wieder auf MODE = 0 rückgesetzt. Wird eine Datenposition angegeben die größer als die aktuelle Dateilänge ist oder die Datei hat die länge 0, so wird bei ERROR 255 ausgegeben (Siehe ERROR Codes bei Baustein FILE_SERVER).

Wenn der Datei-Zugriff nicht mehr benötigt wird, muss vom Anwender entweder durch Nutzung von AUTO_CLOSE oder durch MODE 5 (Datei schließen) über den FILE_SERVER die Datei wieder geschlossen werden.

10.4. FILE_PATH_SPLIT

Type	Funktion : BOOL
INPUT	FILENAME : STRING(STRING_LENGTH)
IN_OUT	X : FILE_PATH_DATA' (einzelne Pfadelemente)

Der Baustein zerlegt einem Dateipfad in seine Einzelelemente. Dabei werden Laufwerk, Pfad und Dateiname extrahiert und in der Datenstruktur X abgelegt. Als Verzeichnistrennzeichen werden „\“ und „/“ akzeptiert. Wenn der übergebene „Filename“ nicht leer ist und Elemente ausgewertet werden können gibt der Baustein TRUE zurück, ansonsten FALSE.

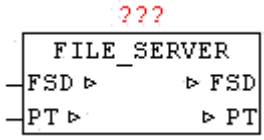
Beispiel:

c:\ordner1\ordner2\oscat.txt

DRIVE DIRECTORY FILENAME

10.5. FILE_SERVER

Type	Funktionsbaustein
IN_OUT	FSD : FILE_SERVER_DATA (Datei Schnittstelle)
	PT: NETWORK_BUFFER (Lese / Schreibdaten)



Verfügbare Plattformen und damit verbundene Abhängigkeiten

CoDeSys:

Benötigt die Bibliothek „SysLibFile.lib“

Lauffähig auf

WAGO 750-841

CoDeSys SP PLCWinNT V2.4

und kompatible Plattformen

PCWORX:

Keine Bibliothek notwendig

Lauffähig auf allen Steuerungen mit File-System ab Firmware >= 3.5x

BECKHOFF:

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0 oder höher	PC or CX (x86)	TcSystem.Lib
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	TcSystem.Lib

Der Baustein FILE_SERVER ermöglicht den Hardware und Hersteller neutralen Zugriff auf das Dateisystem der Steuerung. Da bei fast bei jeder Hardware und Software-Plattform die Zugriffsmöglichkeiten auf das Filesystem mitunter sehr unterschiedlich ausgeprägt sind, ist es dadurch notwendig eine einheitliche vereinfachte und auf die notwendigsten Funktionen reduzierte Schnittstelle zu verwenden. Der Baustein ist Hardware abhängig und somit muss für die jeweilige Plattform immer der passende Implementierung vorliegen.

Mittels FILENAME wird die gewünschte Datei angegeben. Je nach Plattform kann der Syntax leicht unterschiedlich sein (Mit und ohne Pfadangabe). Durch Parameter MODE wird die prinzipielle Zugriffsart vorgegeben. Bei MODE 1,2 und 3 kann mit „OFFSET“ die Position innerhalb der Datei angegeben werden. Beim Dateisystem wird immer mit Byte 0 beginnend gezählt. Im ersten Schritt wird immer geprüft ob diese Datei schon (noch) geöffnet ist, und wenn nicht wird diese geöffnet und die aktuelle Dateigröße wird festgestellt und bei „FILE_SIZE“ ausgegeben. Wird bei AUTO_CLOSE eine Zeit > 0ms angegeben, wird automatisch nach jedem Kommando und nach Ablauf dieser Wartezeit die Datei wieder geschlossen. Alternativ kann mittels MODE = 5 das schließen der Datei manuell durchgeführt werden. Jedes Schreibkommando das die Größe der Datei verändert führt automatisch zu einem korrigierten „FILE_SIZE“ Eintrag, somit ist immer ersichtlich wie groß die Datei momentan ist. Sobald ein Datei geöffnet ist, wird dies über FILE_OPEN = TRUE mitgeteilt.

Sollen Daten gelesen werden, kann über die Datenstruktur PT beim Element SIZE die Anzahl der Bytes angegeben werden, dabei erfolgt eine automatische Anpassung der Datenmenge durch den FILE_SERVER. Je nach Größe des Lese-Buffer und der Menge an lesbaren Datenbytes in der

Datei wird die tatsächliche Leselänge korrigiert und nach erfolgten Lesen der Daten stehen diese in PT.BUFFER zur Verfügung. Dabei wird in PT.SIZE die tatsächliche Datenmenge automatisch korrigiert bzw. eingetragen.

Wird der MODE 1,2 oder 3 mit PT.SIZE = 0 aufgerufen, so wird die Datei geöffnet, die FILE_SIZE bestimmt, jedoch wird kein Lese/Schreib Kommando durchgeführt, und die Datei bleibt geöffnet, bis zum manuellen Schließen bzw. AUTO_CLOSE.

Sollen Daten geschrieben werden, so müssen vor Aufruf, die zu schreibenden Daten in PT.BUFFER und bei PT.SIZE die Anzahl der Bytes eingetragen werden. Diese Daten werden dann Relativ zum angegebene OFFSET in die Datei geschrieben werden. Wird ein Schreib-Mode angerufen bei dem PT.SIZE = 0 ist, dann wird wiederum nur diese Datei geöffnet (wenn noch nicht geöffnet ist, und kein Schreibkommando ausgeführt, und diese bleibt solange geöffnet, bis ein manuellen Schließen oder AUTO_CLOSE durchgeführt wird.

Nach jedem ausgeführten Kommando das die Position des virtuellen Lese / Schreibzeiger verändert , wird die aktuelle Position in der Datenstruktur im Parameter „OFFSET“ eingetragen. Damit kann sehr einfach eine automatische APPEND Funktion realisiert werden. Es muss nur einmalig nach öffnen der Datei der Parameter FILE_SIZE nach OFFSET übertragen werden. Danach werden alle geschriebenen Bytes immer an Ende angehängt ohne das der Parameter OFFSET manuell verändert werden muss. Gleiches Prinzip kann beim Lesen angewandt werden, dabei sollte natürlich der Lesezeiger zuerst innerhalb der Datei positioniert werden (normalerweise beginnend bei OFFSET 0).

Wird ein Kommando ausgeführt und FILENAME unterscheidet sich vom aktuellen FILENAME , so wird die alte noch offene Datei automatisch geschlossen, und die neue danach geöffnet, und mit dem normalen Kommando fortgesetzt. Damit lässt sich ohne großen Aufwand ein fliegender Wechsel der Datei durchführen, ohne umständlich vorher CLOSE und OPEN durchführen zu müssen.

Beim Löschen einer Datei mit MODE 4 wird automatisch eine möglicherweise noch offene Datei vorher geschlossen, und dann in Folge gelöscht.

Nach einem AUTO_CLOSE oder manuellen Schließen durch MODE 5 werden nach Durchführung der Aktion in FILE_SERVER_DATA alle Informationen rückgesetzt.

Der Baustein FILE_SERVER sollte immer zyklisch aufgerufen werden , zumindest solange nicht alle Zugriffe sicher beendet sind.

Da auf manchen Plattformen die Dateizugriffe zum Teil blockierend arbeiten (z.B. CoDeSys) und nicht immer eine asynchrone Nutzung erlauben, sollten der FILE_SERVER möglichst in einem eigenen Task laufen, damit die Standard-Applikation im Zeitverhalten nicht zu beeinflussen.

Der FILE_SERVER stellt folgenden Befehle über „MODE“ zur Verfügung

MODE	Eigenschaften
1	Eine vorhandene Datei wird für den Lesezugriff geöffnet und Daten werden optional gelesen
2	Eine vorhandene Datei wird für den Schreibzugriff geöffnet und Daten werden optional geschrieben
3	Eine Datei wird für den Schreibzugriff neu angelegt und Daten werden optional geschrieben
4	Datei löschen
5	Datei schließen

ERROR: Fehlercodes Beckhoff

Wert	Auslöser	Beschreibung
0		Kein Fehler
19	SYSTEMSERVICE_FOPEN	Unbekannte oder ungültige Parameter
28	SYSTEMSERVICE_FOPEN	Datei nicht gefunden. Ungültiger Dateiname oder Dateipfad
38	SYSTEMSERVICE_FOPEN	Keine weiteren freien File-Handle.
51	SYSTEMSERVICE_FCLOSE	ungültiges oder unbekanntes File-Handle.
62	SYSTEMSERVICE_FCLOSE	Datei wurde mit falscher Methode geöffnet.
67	SYSTEMSERVICE_FREAD	ungültiges oder unbekanntes File-Handle.
74	SYSTEMSERVICE_FREAD	Kein Speicher für Lesebuffer.
78	SYSTEMSERVICE_FREAD	Datei wurde mit falscher Methode geöffnet.
83	SYSTEMSERVICE_FWRITE	ungültiges oder unbekanntes File-Handle
94	SYSTEMSERVICE_FWRITE	Datei wurde mit falscher Methode geöffnet.
99	SYSTEMSERVICE_FSEEK	ungültiges oder unbekanntes File-Handle.
110	SYSTEMSERVICE_FSEEK	Datei wurde mit falscher Methode geöffnet.
115	SYSTEMSERVICE_FTELL	ungültiges oder unbekanntes File handle.
126	SYSTEMSERVICE_FTELL	Datei wurde mit falscher Methode geöffnet
140	SYSTEMSERVICE_FDELETE	Datei nicht gefunden. Ungültiger Dateiname oder Dateipfad.
255	Applikation	Position liegt hinter dem Dateiende

ERROR: Fehlercodes PCWORX:

Wert	Auslöser	Beschreibung
0		Kein Fehler
2	FILE_OPEN	Die maximale Anzahl von Dateien ist bereits geöffnet
4	FILE_OPEN	Die Datei ist bereits geöffnet
5	FILE_OPEN	Die Datei ist schreibgeschützt oder Zugriff verweigert
6	FILE_OPEN	Dateiname nicht angegeben
11	FILE_CLOSE	Ungültiger Datei handle
30	FILE_CLOSE	Datei konnte nicht geschlossen werden
41	FILE_READ	Ungültiger Datei handle
50	FILE_READ	Dateiende erreicht
52	FILE_READ	Die Anzahl der zu lesenden Zeichen ist größer als der Datenpuffer
62	FILE_READ	Es konnten keine Daten gelesen werden
71	FILE_WRITE	Ungültiger Datei handle
81	FILE_WRITE	Es ist kein Speicher zum Schreiben der Daten verfügbar
82	FILE_WRITE	Die Anzahl der zu schreibenden Zeichen ist größer als der Datenpuffer
93	FILE_WRITE	Es konnten keine Daten geschrieben werden
0	FILE_SEEK	Ungültiger Datei handle
113	FILE_SEEK	Ungültiger Positionierungsmodus oder die angegebene Position liegt vor dem Dateianfang
124	FILE_SEEK	Die Position konnte nicht gesetzt werden
131	FILE_TELL	Ungültiger Datei handle
142	FILE_REMOVE	Die maximale Anzahl von Dateien ist bereits geöffnet
143	FILE_REMOVE	Die Datei konnte nicht gefunden werden
145	FILE_REMOVE	Die Datei ist geöffnet, schreibgeschützt oder Zugriff verweigert
146	FILE_REMOVE	Dateiname nicht angegeben
161	FILE_REMOVE	Datei konnte nicht gelöscht werden
255	Applikation	Position liegt hinter dem Dateiende

ERROR: Fehlercodes CoDeSys:

Wert	Auslöser	Beschreibung
0		Kein Fehler
1	SysFileOpen	Fehler
2	SysFileClose	Fehler
3	SysFileRead	Fehler
4	SysFileWrite	Fehler
5	SysFileSetPos	Fehler
6	SysFileGetPos	Fehler
7	SysFileDelete	Fehler
8	SysFileGetSize	Fehler
255	Applikation	Position liegt hinter dem Dateiende

10.6. INI-Dateien

Eine Initialisierungsdatei (kurz INI-Datei) ist eine Textdatei, die z.B. unter Windows zum Ablegen von Programmeinstellungen (z. B. Position des Programmfensters) verwendet wird. Bei erneutem Aufruf des Programms werden die Programmeinstellungen eingelesen, um den Zustand vor dem letzten Schließen wieder einzunehmen.

Aufgrund der sehr einfachen funktionellen Strukturierung und Handhabung bietet sich dieser quasi Standard an, auch für Programmeinstellungen und ähnliches bei Steuerungen mit Filesystem einzusetzen.

Eine INI-Datei kann in Sektionen unterteilt werden, welche mit eckigen Klammern umschlossen sein müssen. Informationen werden als Schlüssel mit zugehörigem Wert ausgelesen.

Bei dem Erstellen einer INI-Datei sind folgende Regeln zu beachten:

Jede Sektion darf nur einmal vorkommen.

Jeder Schlüssel darf nur einmal je Sektion vorkommen.

Auf Werte wird mittels Sektion und Schlüssel zugegriffen.

Eine Sektion kann auch keinen Schlüssel enthalten

Kommentare werden mit einem "#" eingeleitet

Kommentare dürfen nicht direkt hinter einem Schlüssel oder einer Sektion stehen.

Kommentare müssen immer in einer neuen Zeile beginnen

Wird bei einem Schlüssel kein Wert angegeben, so wird ein Leerstring als Wert gemeldet.

Jede Sektion und jeder Schlüssel bzw. folgender Wert müssen mit einen Zeilenumbruch abgeschlossen werden. Dabei spielt die Art des Zeilenumbruch Zeichens keine Rolle, da alle Varianten akzeptiert werden. Häufigste Variante ist <CR><LF>. Es werden aber alle Steuerzeichen (nicht darstellbare Zeichen) als Zeilenende interpretiert.

Leerzeichen werden immer als Teile der Elemente betrachtet und auch so ausgewertet

Es können prinzipiell beliebig viele Sektion und Schlüssel verwendet werden.

Prinzipieller Aufbau:

#Kommentar<CR><LF>
[Sektion]<CR><LF>
#Kommentar<CR><LF>
Schlüssel=Wert<CR><LF>

Beispiel:

```
[SYSTEM]
DEBUG_LEVEL=10
QUIT_TIME=5
```

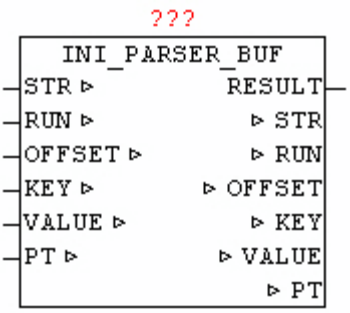
```
#--------------------------
# Station 1 Parametrierung -
#--------------------------
[Station_1]
NAME=ILC150 ETH
IP=192.168.15.100
M2=S2/M3/C1
#--------------------------
# Station 2 Parametrierung -
#--------------------------
[Station_2]
NAME=ILC350PN
IP=192.168.15.108
M1=S1/M1
M2=S3/M2
```

10.7. INI_PARSER_BUF

Type Funktionsbaustein

OUTPUT: RESULT : BYTE (Ergebnis der Abfrage)

IN_OUT STR : STRING(STRING_LENGTH) (gesuchtes Elements)
RUN : BYTE (Befehlscode für aktuelle Aktion)
OFFSET : UDINT (aktueller Datei-Offset der Abfrage)
KEY : STRING(STRING_LENGTH) (gefundenes Elements)
VALUE : STRING(STRING_LENGTH) (Wert eines Schlüssels)
PT: NETWORK_BUFFER (Lese Datenbuffer)



Der Baustein INI_PARSER_BUF ermöglicht die Auswertung der Elemente einer INI-Datei die in einem Byte-Array abgelegt sind. Bevor Abfragen durchgeführt werden können muss vom Anwender das Byte-Array PT.BUFFER mit den INI-Daten gefüllt werden, und die Anzahl der Bytes in PT.SIZE eingetragen werden. Die Suche nach Elementen beginnt immer abhängig vom übergebenen „OFFSET“, somit ist sehr leicht möglich nur an bestimmten Stellen zu suchen, bzw. die Suche ab einer bestimmten Section zu wiederholen, um nicht immer das gesamte Byte-Array durchsuchen zu müssen. Bei der erstmaligen Suche sollte Standardmäßig mit OFFSET 0 begonnen werden (muss aber nicht !). Bei der Abfrage von Sections und Keys gibt es verschiedene Vorgangsweisen. Entweder man sucht gezielt nach einer Section und wertet alle nachfolgenden Keys durch einzelne Abfragen aus, oder man benutzt bei sehr umfangreichen Initialisierungsdateien die klassische Enumeration (Auflistung), das heißt es werden seriell alle Elemente gemeldet, und von der Applikation weiterverarbeitet.

Section suchen:

Um den OFFSET einer bestimmten Section zu bestimmen, muss bei STR der Name der Section angegeben werden, sowie der OFFSET auf eine Position gesetzt werden, die logischer Weise vor der zu suchenden Section liegt. Soll keine bestimmte, sonder nur die nächste vorhandene Section gefunden werden, so muss bei STR ein leerer String übergeben werden. Die Suchabfrage wird durch RUN = 1 gestartet. Die Suche dauert je nach Aufbau und Größe der INI-Daten unterschiedlich lange, dass heißt es dauert eine unbestimmte Anzahl an Zyklen lang, bis ein positives oder negatives Ergebnis vorliegt. Sobald die Suche beendet ist, wird vom INI_PARSER_BUF der Parameter RUN auf 0 gesetzt. Über RESULT wird das Ergeb nis der Suche ausgegeben. Bei erfolgreicher suche wird dann beim Parameter KEY der Name der gefunden Section ausgegeben. Sowie zeigt dann der Parameter OFFSET auf das Ende der Sections-Zeile. Somit kann direkt im Anschluss mit der Key Auswertung fortgesetzt werden, ohne den OFFSET manuell verändern zu müssen.

Key suchen:

Um einen Key auszuwerten, muss zuvor der OFFSET auf einen korrekten Wert eingestellt werden, dies kann durch manuelle OFFSET Vorgabe oder durch eine zuvor durchgeführte Section suche durchgeführt werden.Vor Beginn der Abfrage muss bei STR der Name des Key übergeben werden, wird ein leerer String bei STR übergeben, so wird der nächste gefundene Key zurückgegeben. Mittels RUN = 2 kann die Abfrage gestartet werden. Sobald die Suche beendet ist, wird vom INI_PARSER_BUF der Parameter RUN auf 0 gesetzt. Über RESULT wird das Ergebnis der Suche ausgegeben. Wird bei der suche nach einem Key der Beginn einer neuen Section erkannt, wird dies durch RESULT = 11 gemeldet. Bei erfolgreicher suche wird dann beim Parameter KEY der Name des gefundenen Schlüssels ausgegeben, und bei VALUE der Schlüssel-Wert. Sowie zeigt dann der Parameter OFFSET auf das Ende der Key-Zeile. Somit kann direkt im Anschluss mit der nächsten Key Auswertung fortgesetzt werden, ohne den OFFSET manuell verändern zu müssen.

Enumeration – nächstes Element finden:

Bei sehr großer Menge an auszuwertenden Daten einer Initialisierungsdatei kann mittels Enumeration (Auflistung) das Anwender-Programm einfach aufgebaut werden, und die Auswertung auch schneller durchgeführt werden, die hier keine Zeile mehr als einmal durchlaufen werden muss. Vor dem Beginn muss OFFSET auf einem logisch vernünftigen Wert gesetzt werden, im Standardfall auf 0. Mittels RUN = 3 wird die Auswertung gestartet. Sobald eine Section oder ein Key gefunden wird, wird dieser auch sofort ausgegeben. Bei einer Section wird bei KEY der Name der Section ausgegeben und RESULT = 1. Bei einem gefundenem KEY wird bei KEY der Key-Name und bei VALUE der Schlüssel-Wert ausgegeben, sowie bei RESULT = 2

Wird bei einer Abfrage das Ende des Daten-Arrays erreicht, so wird dies mittels RESULT = 10 gemeldet.

RUN : Funktionsübersicht

RUN	Funktion
0	Keine Funktion durchführen – bzw. letzte Funktion wurde beendet
1	Bestimmte Sektion bzw. nächste gefundene Sektion auswerten
2	Bestimmten Key bzw. nächsten gefundenen Key auswerten
3	Nächstes gefundene Element (Section oder Key) auswerten

RESULT : Ergebnis - Rückmeldung

RESULT	Beschreibung
1	Section gefunden
2	Key gefunden
5	Aktuelle Abfrage läuft noch – Baustein weiter zyklisch aufrufen !
10	Nichts gefunden – Daten-Ende erreicht
11	Keinen Key gefunden – Ende der Section erreicht

10.8. INI_PARSER_FILE

Type Funktionsbaustein

OUTPUT: RESULT : BYTE (Ergebnis der Abfrage)

IN_OUT FILENAME : STRING (Dateiname)

FSD : FILE_SERVER_DATA (Datei Schnittstelle)

STR : STRING(STRING_LENGTH) (gesuchtes Elements)

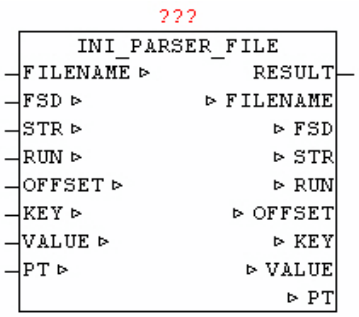
RUN : BYTE (Befehlscode für aktuelle Aktion)

OFFSET : UDINT (aktueller Datei-Offset der Abfrage)

KEY : STRING(STRING_LENGTH) (gefundenes Elements)

VALUE : STRING(STRING_LENGTH) (Wert eines Schlüssels)

PT: NETWORK_BUFFER (Lese Datenbuffer)



Der Baustein INI_PARSER_FILE ermöglicht die Auswertung der Elemente einer beliebig großen INI-Datei die zur Verarbeitung automatisch

Blockweise in den Lese Datenbuffer eingelesen wird. Der Name der Datei wird bei Parameter „FILENAME“ übergeben. Die Suche nach Elementen beginnt immer abhängig vom übergebenen „OFFSET“, somit ist sehr leicht möglich nur an bestimmten Stellen zu suchen, bzw. die Suche ab einer bestimmten Section zu wiederholen, um nicht immer die gesamte Datei durchsuchen zu müssen. Bei der erstmaligen Suche sollte Standardmäßig mit OFFSET 0 begonnen werden (muss aber nicht !). Bei der Abfrage von Sections und Keys gibt es verschiedene Vorgangsweisen. Entweder man sucht gezielt nach einer Section und wertet danach die gewünschten Keys durch einzelne Abfragen aus, oder man benutzt bei sehr umfangreichen Initialisierungsdateien die klassische Enumeration (Auflistung), das heißt es werden seriell alle Elemente gemeldet, und von der Applikation weiterverarbeitet.

Section suchen:

Um den OFFSET einer bestimmten Section zu bestimmen, muss bei STR der Name der Section angegeben werden, sowie der OFFSET auf eine Position gesetzt werden, die logischer Weise vor der zu suchenden Section liegt. Soll keine bestimmte, sonder nur die nächste vorhandene Section gefunden werden, so muss bei STR ein leerer String übergeben werden. Die Suchabfrage wird durch RUN = 1 gestartet. Die Suche dauert je nach Aufbau und Größe der INI-Daten unterschiedlich lange, dass heißt es dauert eine unbestimmte Anzahl an Zyklen lang, bis ein positives oder negatives Ergebnis vorliegt. Sobald die Suche beendet ist, wird vom INI_PARSER_BUF der Parameter RUN auf 0 gesetzt. Über RESULT wird das Ergeb nis der Suche ausgegeben. Bei erfolgreicher suche wird dann beim Parameter KEY der Name der gefunden Section ausgegeben. Sowie zeigt dann der Parameter OFFSET auf das Ende der Sections-Zeile. Somit kann direkt im Anschluss mit der Key Auswertung fortgesetzt werden, ohne den OFFSET manuell verändern zu müssen.

Key suchen:

Um einen Key auszuwerten, muss zuvor der OFFSET auf einen korrekten Wert eingestellt werden, dies kann durch manuelle OFFSET Vorgabe oder durch eine zuvor durchgeführte Section suche durchgeführt werden.Vor Beginn der Abfrage muss bei STR der Name des Key übergeben werden, wird ein leerer String bei STR übergeben, so wird der nächste gefundene Key zurückgegeben. Mittels RUN = 2 kann die Abfrage gestartet werden. Sobald die Suche beendet ist, wird vom Baustein der Parameter RUN auf 0 gesetzt. Über RESULT wird das Ergebnis der Suche ausgegeben. Wird bei der suche nach einem Key der Beginn einer neuen Section erkannt, wird dies durch RESULT = 11 gemeldet. Bei erfolgreicher suche wird dann beim Parameter KEY der Name des gefundenen Schlüssels ausgegeben, und bei VALUE der Schlüssel-Wert. Sowie zeigt dann der Parameter OFFSET auf das Ende der Key-Zeile. Somit kann direkt im Anschluss mit der nächsten Key

Auswertung fortgesetzt werden, ohne den OFFSET manuell verändern zu müssen.

Enumeration – nächstes Element finden:

Bei sehr großer Menge an auszuwertenden Daten einer Initialisierungsdatei kann mittels Enumeration (Auflistung) das Anwender-Programm einfach aufgebaut werden, und die Auswertung auch schneller durchgeführt werden, die hier keine Zeile mehr als einmal durchlaufen werden muss. Vor dem Beginn muss OFFSET auf einem logisch vernünftigen Wert gesetzt werden, im Standardfall auf 0. Mittels RUN = 3 wird die Auswertung gestartet. Sobald eine Section oder ein Key gefunden wird, wird dieser auch sofort ausgegeben. Bei einer Section wird bei KEY der Name der Section ausgegeben und RESULT = 1. Bei einem gefundenem KEY wird bei KEY der Key-Name und bei VALUE der Schlüssel-Wert ausgegeben, sowie bei RESULT = 2

Wird bei einer Abfrage das Ende des Daten-Arrays erreicht, so wird dies mittels RESULT = 10 gemeldet.

Wenn der Datei-Zugriff nicht mehr benötigt wird, muss vom Anwender entweder durch Nutzung von AUTO_CLOSE oder durch MODE 5 (Datei schließen) über den FILE_SERVER die Datei wieder geschlossen werden.

RUN : Funktionsübersicht

RUN	**Funktion**
0	Keine Funktion durchführen – bzw. letzte Funktion ist beendet
1	Bestimmte Sektion bzw. nächste Sektion auswerten
2	Bestimmten Key bzw. nächsten Key auswerten
3	Nächstes gefundene Element (Section oder Key) auswerten

RESULT : Ergebnis - Rückmeldung

RESULT	**Beschreibung**
1	Section gefunden
2	Key gefunden
5	Aktuelle Abfrage läuft noch – Baustein weiter zyklisch aufrufen !
10	Nichts gefunden – Daten-Ende erreicht

11	Keinen Key gefunden – Ende der Section erreicht

11. Telnet-Vision

11.1. TELNET_VISION

Der Paket TELNET_VISION ist ein Framework das eine Vielzahl an Funktionsbausteinen umfasst, um mit einfachen Mitteln eine grafische Oberfläche auf Basis des Standards TELNET zu ermöglichen.

Das GUI (Grafik-User-Interface) benutzt einen Bildschirm der 80 Zeichen Breit und 24 Zeilen hoch ist. An jeder Koordinate (Position) kann ein beliebiges darstellbares Zeichen mit wählbaren Farbe-Attributen ausgegeben werden.

Die Horizontale Achse (von links nach rechts) wird standardmäßig mit X Bezeichnet und umfasst die Positionen 00-79. Die Vertikale Achse (von oben nach unten) wird standardmäßig mit Y Bezeichnet und umfasst die Positionen 00-23. Bei reinen Koordinaten wird die Position mit X und Y angeben. Wird eine Fläche (Rechteck) angegeben, so wird die linke obere Ecke mit X1/Y1 und die rechte untere Ecke mit X2,Y2 definiert.

Die einzelnen Zeichen können mit Farb-Attributen versehen werden. Ein Farb-Attribute bestehen aus einem Byte, wobei das linke Nibble (4Bit) die Schreibfarbe (Vordergrundfarbe) und das rechte Nibble (4Bit) die Hintergrundfarbe definiert.

Beispiel : BYTE#16#74, (* Vordergrund: Weiss , und Hintergrund Blau *)

Folgende Farb-Attribute sind definiert:

Vordergrundfarbe:

Nibble	Farbe	Byte	Farbe
0	Black	8	Flashing Black
1	Light Red	9	Flashing Light Red
2	Light Green	10	Flashing Light Green
3	Yellow	11	Flashing Yellow
4	Light Blue	12	Flashing Light Blue
5	Pink / Light Magenta	13	Flashing Pink / Light Magenta
6	Light Cyan	14	Flashing Light Cyan

7	White	15	Flashing White

Hintergrundfarbe:

Nibble	Farbe
0	Black
1	Red
2	Green
3	Brown
4	Blue
5	Purple / Magenta
6	Cyan
7	Gray

Für die einfache Handhabung des Paketes ist der Baustein TN_FRAMEWORK zuständig, dieser muss in der Applikation zyklisch aufgerufen werden, da dieser das ganze System verwaltet und ausführt. Dabei wird die Kommunikation mit dem Telnet-Client durchgeführt, bei grafisches Änderungen wird vom System immer eine intelligente automatische Aktualisierung durchgeführt. Auch die INPUT_CONTROL Elemente werden verwaltet, sowie die Tastatureingaben an die jeweiligen Elemente weitergeleitet, und auch eine optionale Menu-Bar ist nutzbar.

Das es sich hierbei um eine relativ komplexes Zusammenspiel von vielen Bausteine handelt, wurden in der Bibliothek unter /DEMO zwei Anwendungsbeispiele zur Verfügung gestellt, die alle Möglichkeiten vorführen. Es ist zu Empfehlen das eigene Projekte auf Basis dieser beiden Vorlagen erstellt werden, um möglichst schnell ein lauffähiges Ergebnis zu bekommen, und um das Zusammenspiel der einzelnen Komponenten und Bausteine zu verstehen.

Das Programm TN_VISION_DEMO_1 zeigt folgende Elemente:

Grafische Darstellung von Linien,Flächen, Texten, und zugehörigen Schatten, sowie farbliche Gestaltung des Layouts

Darstellung einer Menu-Bar

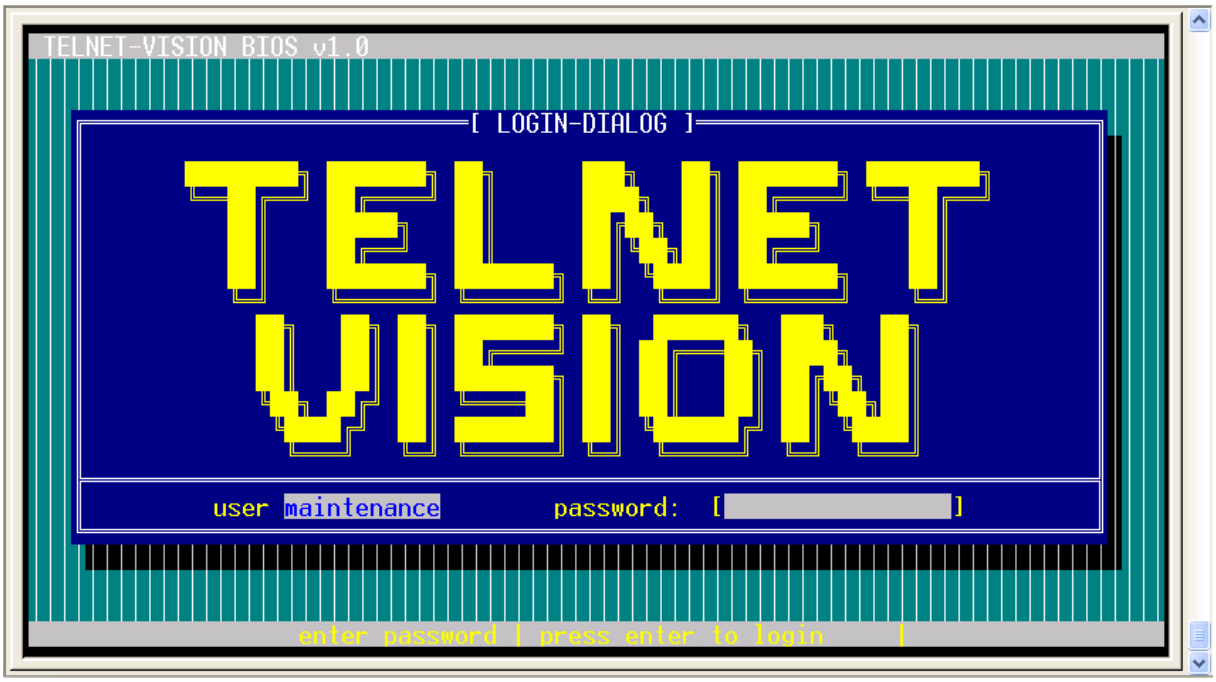
Elemente: EDIT_LINE (normale und verdeckte Eingabe), SELECT_POPUP, SELECT_TEXT

TOOLTIP Infozeile

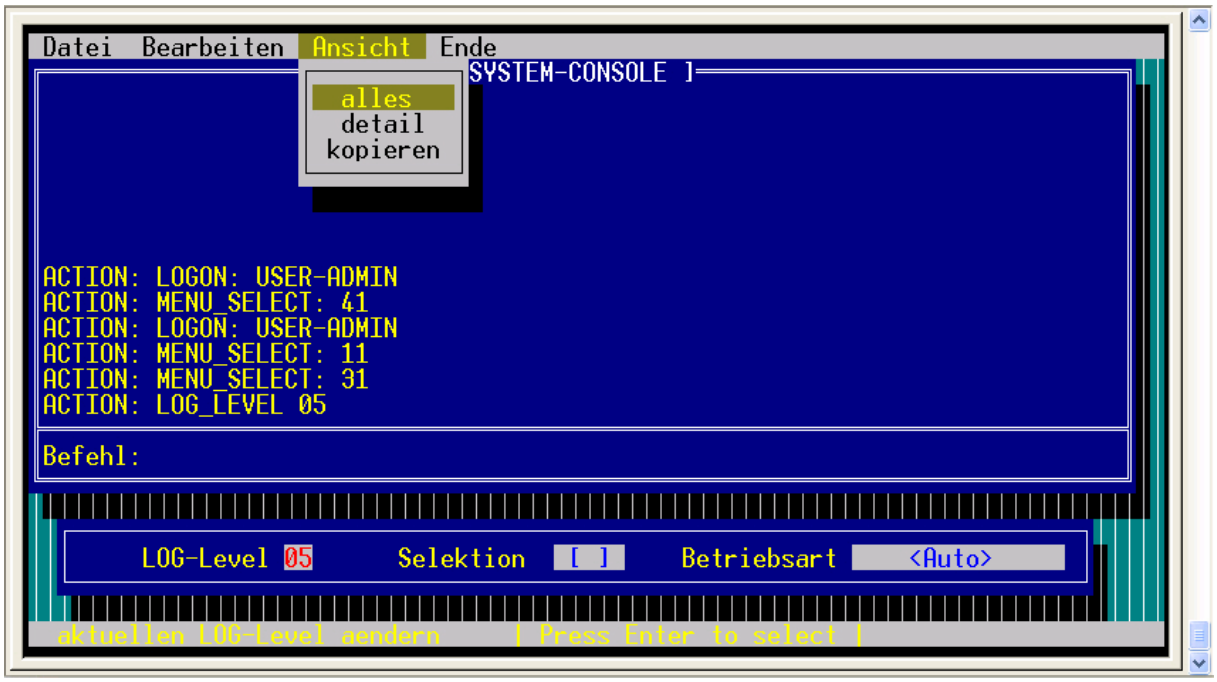
Darstellung eines LOG_VIEWPORT in dem der Meldungspuffer dargestellt wird, und man mittels Tasten navigieren kann.

Auf der Startseite ist eine LOGIN-Funktion realisiert. Durch Eingabe des Passwortes 'oscat' kann man auf die Folgeseite wechseln. Auf der Hauptseite kann dann mit den Cursor Auf/Ab und Tabulator Taste zwischen den einzelnen Elemente gewechselt werden. Das Menu kann mit der Taste Escape aufgerufen werden. Die einzelnen Menu-Punkte sind nur zu Demonstrationszwecke und führen nur zu einen LOG-Meldung. Lediglich der Menu-Punkt „Ende/LOGOUT“ führt wieder zur Startseite zurück.

TN_VISION_DEMO_1 (Bildschirmseite 1)



TN_VISION_DEMO_1 (Bildschirmseite 2)



Das Programm TN_VISION_DEMO_2 zeigt folgende Elemente:

Grafische Darstellung von Linien,Flächen, Texten, sowie monchrome Gestaltung des Layouts

Elemente: EDIT_LINE (normale und verdeckte Eingabe, sowie Nutzung einer Eingabemaske), SELECT_TEXT

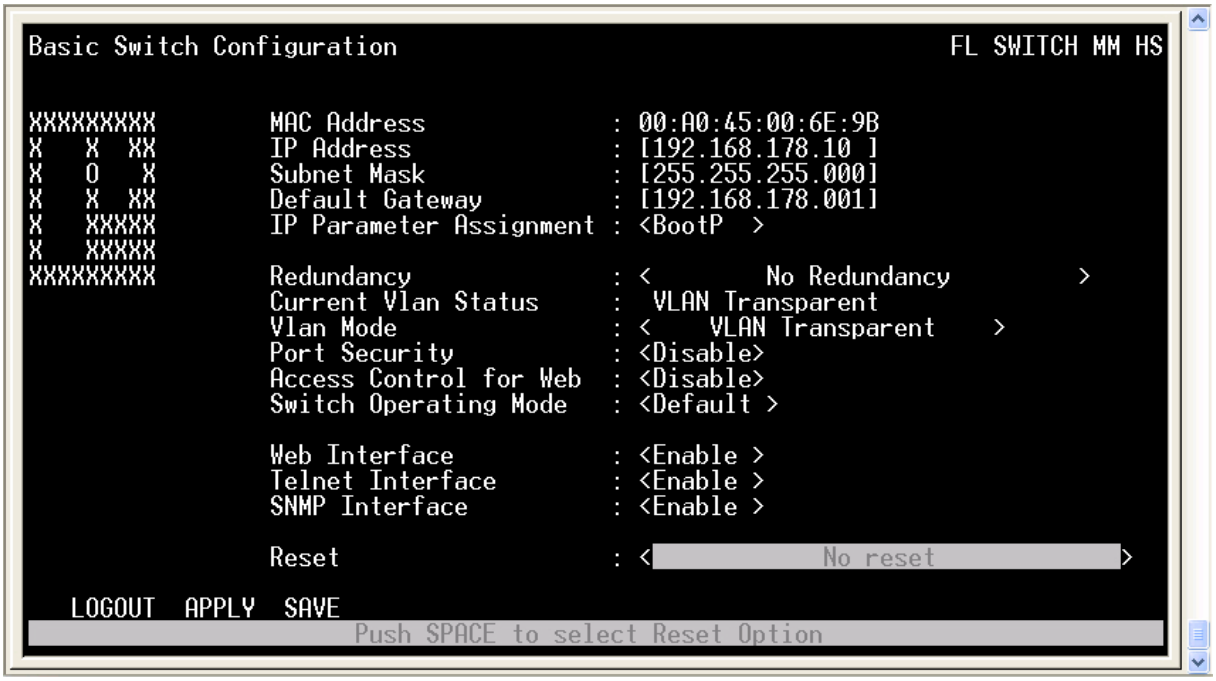
TOOLTIP Infozeile

Auf der Startseite ist eine LOGIN-Funktion realisiert. Durch Eingabe des Passwortes 'oscat' kann man auf die Folgeseite wechseln. Auf der Hauptseite kann dann mit den Cursor Auf/Ab und Tabulator Taste zwischen den einzelnen Elemente gewechselt werden. Lediglich das Element „LOGOUT“ führt wieder zur Startseite zurück.

Die dargestellten zwei Seiten sind eine Nachbildung der Telnet-Seiten eines Managebaren Switch der Fa. PHOENIX-CONTACT, um zu zeigen das man das TELNET-VISION Paket auch für einfache Konfigurationsseiten nutzen kann.

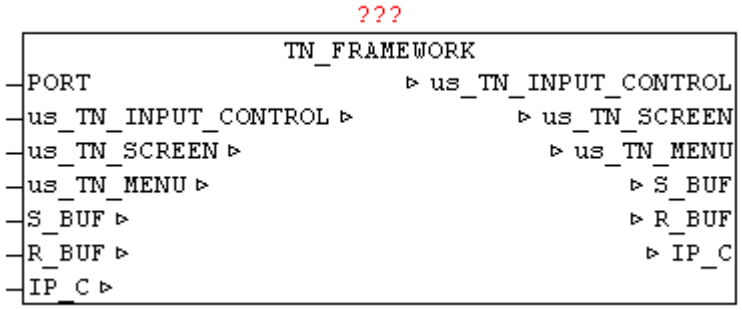
TN_VISION_DEMO_2 (Bildschirmseite 1)

TN_VISION_DEMO_2 (Bildschirmseite 2)



11.2. TN_FRAMEWORK

Type	Funktionsbaustein
IN	PORT : WORD (optionale PORT-Nummer)
IN_OUT	Xus_TN_INPUT_CONTROL : us_TN_INPUT_CONTROL
	Xus_TN_SCREEN : us_TN_SCREEN
	Xus_TN_MENU : us_TN_MENU
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
	IP_C : IP_CONTROL (Parametrierungsdaten)



Der Baustein TN_FRAMEWORK ist eine Rahmenstruktur die ein fertiges Laufzeitmodell für TELNET-Vision zur Verfügung stellt.

Der Parameter PORT ermöglicht die optionale Vorgabe der PORT Nummer. Wird der Parameter nicht vorgegeben, so wird der Standard Port 23 benutzt. Alle anderen Parameter sind reine Datenstrukturen die für die interne Funktion des Framework notwendig sind.

Folgende Aufgaben und Funktionen werden abgehandelt.

Verbindungsaufbau und Abbau mit Telnet-Client

Daten senden und Empfangen

Datenstrukturen für grafische Funktionen

Datenstrukturen für INPUT_CONTROL Elemente

Automatisches Intelligentes Updaten des Telnet-Anzeige

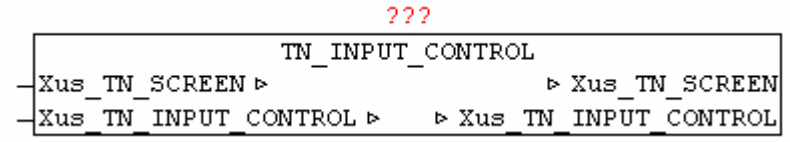
Menu-Bar Darstellung

Direkter Zugang zu allen Datenstrukturen für Anwenderprogramm

11.3. TN_INPUT_CONTROL

Type Funktionsbaustein

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN
Xus_TN_INPUT_CONTROL : us_TN_INPUT_CONTROL



Der Baustein TN_INPUT_CONTROL dient zum verwalten der INPUT_CONTROL Elemente. Wird Xus_TN_INPUT_CONTROL.bo_Reset_Fokus = TRUE dann wird bei allen Elementen der Fokus deaktiviert und das erste Element bekommt den Fokus zugeteilt. Mit den Tasten Cursor Auf/Ab und Tabulator können die einzelnen Elemente angewählt bzw. gewechselt werden. Das aktuelle Element verliert dabei den Fokus und das nächstfolgende Element bekommt danach den Eingabe-Fokus neu zugeteilt. Bei dem Fokus Wechsel der Elemente wird immer automatisch das neu zeichnen der jeweiligen Elemente ausgelöst. Der Grafik / Blink-Cursor wird dabei immer je aktiven Element positioniert und dargestellt. Dabei wird auch immer automatisch der ToolTip Text ausgegeben und aktualisiert, soweit dieser parametriert wurde.

Es werden folgende Elemente unterstützt.

TN_INPUT_EDIT_LINE

TN_INPUT_SELECT_TEXT

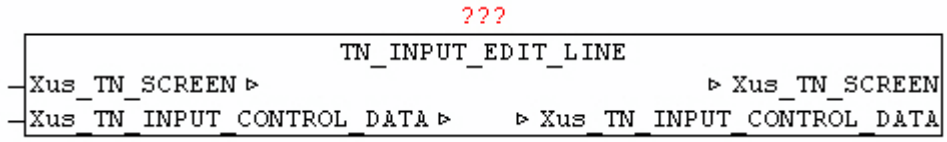
TN_INPUT_SELECT_POPUP

11.4. TN_INPUT_EDIT_LINE

Type Funktionsbaustein

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN

Xus_TN_INPUT_CONTROL : us_TN_INPUT_CONTROL



Der Baustein TN_INPUT_EDIT_LINE dient zum verwalten einer Eingabezeile. Dazu muss *.in_Type = 1 gesetzt werden.

Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.

Besitzt das Element den Fokus , kann mittels der Tasten Cursor Links/Rechts innerhalb der Zeile der Blink-Cursor verschoben werden. Mit der Backspace Taste können eingegebene Zeichen wieder gelöscht werden. Durch Betätigen der Eingabe/Return Taste wird der Eingabetext bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden. Mittels *.bo_Input_Hidden = TRUE wird die verdeckte Eingabe aktiviert, dadurch werden alle eingegebenen Zeichen mit '*' dargestellt.

Mittels *.st_Input_Mask wird bestimmt an welcher Position und wie viele Zeichen eingegeben werden können. An jeder Position an der sich ein Leerzeichen befindet können eingaben gemacht werden. Bei der Initialisierung muss einmalig *.st_Input_Mask nach *.st_Input_Data kopiert werden.

Ist *.bo_Input_Only_Num = TRUE werden nur nummerische Tasten akzeptiert und übernommen.

Beispiel:

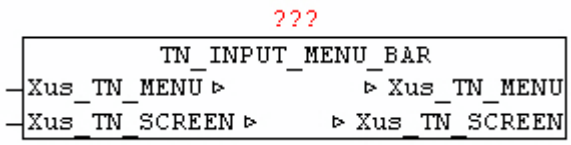
```
*.in_Type := INT#01;
*.in_Y := INT#16;
*.in_X := INT#09;
*.by_Attr_mF := BYTE#16#72; (* Weiss, Grün *)
*.by_Attr_oF := BYTE#16#74; (* Weiss, Blau *)
*.in_Cursor_Pos := INT#0;
*.bo_Input_Only_Num := FALSE;
*.bo_Input_Hidden := FALSE;
*.st_Input_Mask := '                                                  ';
*.st_Input_Data :=  *.st_Input_Mask;
*.st_Input_ToolTip := 'Eingabezeile aktiv       | SCROLL F1/F2/F3/F4 |';
*.in_Input_Option := INT#02;
```

```
*.in_Title_Y_Offset := INT#00;
*.in_Title_X_Offset := INT#00;
*.by_Title_Attr := BYTE#16#34;
*.st_Title_String := 'Befehl: ';
```

Ergibt folgende Ausgabe:

11.5. TN_INPUT_MENU_BAR

Type	Funktionsbaustein
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN
	Xus_TN_MENU : us_TN_MENU



Der Baustein TN_INPUT_MENU_BAR dient zum verwalten und anzeigen der Menu_Bar. Das Element wird an *.in_X und *.in_Y dargestellt. Die Menu-Elemente sind als verschachelte Elemente innerhalb *.st_MENU_TEXT hinterlegt. Dabei werden zwei verschiedene Trennzeichen verwendet. Ein '$' trennt die verschiedenen Menu-Listen, und jede Menu-Liste wird wiederum mittels '#' in einzelne Menu-Elemente unterteilt. Die erste Menu-Liste stellt die eigentliche Menu-Bar dar, das heißt daraus ergibt sich die Anzahl der Sub-Menu bzw. die Überschriften der Elemente. Danach folgen alle Sub-Menu-Listen immer getrennt mit '%'. Um einzelne Sub-Menu Elemente voneinader abzugrenzen bzw. mit einer Trennlinie zu versehen, muss als Text des Menu-Elementes ein '-' angegeben werden.

Durch betätigen er Escape-Taste wird die Menu-Bar aktiviert und das jeweilige Sub-Menu wird mit Hilfe des Bausteins TN_INPUT_MENU_POPUP dargestellt. Innerhalb des Sub-Menu kann mit Cursor oben/unten navigiert werden. Wird Cursor Links/Rechts betätigt so wird automatisch zum nächsten Haupt-Menu gewechselt und das zugehörige Sub-Menu

angezeigt. Wird ein Sub-Menu-Element mit Enter/Return Taste bestätigt, so wird bei *.in_Menu_Selected die Nummer des gewählten Menu-Punktes ausgegeben. Die Berechnung der Menu-Punktnummer erfolgt folgend: Hauptmenu-Index * 10 + Submenu-Index. Der Eintrag in *.in_Menu_Selected muss vom Anwender nach Entgegennahme wieder auf 0 gesetzt werden.

Somit sind maximal 9 Hauptmenu-Items und je 9 Submenu-Items ausführbar. Mittels Escape-Taste kann jederzeit das Menu wieder ausgeblendet werden.

Ein aktives Menu sichert automatisch den Hintergrund bevor es gezeichnet wird, und restauriert den Hintergrund wieder nach Beendigung.

Solange eine Menu-Anzeige aktiv ist, dürfen vom Anwenderprogramm aber keinerlei grafisches Veränderungen gemacht werden. Dies kann mittels TN_SCREEN.bo_Menue_Bar_Dialog = TRUE überprüft werden.

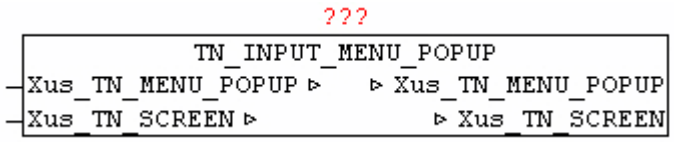
Beispiel:

```
*.in_X := INT#00;
*.in_Y := INT#00;
*.by_Attr_mF := BYTE#16#33; (* yellow + brown *)
*.by_Attr_oF := BYTE#16#0F; (* schwarz + grau *)
*.st_MENU_TEXT := 'Datei#Bearbeiten#Ansicht#Ende';
*.st_MENU_TEXT := CONCAT(*.st_MENU_TEXT,
'%oeffnen#-#speichern#beenden%loeschen#-#einfuegen#-#kopieren');
*.st_MENU_TEXT := CONCAT(*.st_MENU_TEXT,
'%alles#detail#kopieren%Logout');
*.bo_Create := TRUE;
```

Ergibt folgende Ausgabe:

11.6. TN_INPUT_MENU_POPUP

Type	Funktionsbaustein
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN
	Xus_TN_MENU : us_TN_MENU



Der Baustein TN_INPUT_MENU_POPUP dient zum verwalten und anzeigen der Menu_Bar Submenu und für der Darstellung von TN_INPUT_SELECT_POPUP Elementen. Das Element wird an *.in_X und *.in_Y dargestellt. Die Menu-Elemente sind als Elemente innerhalb *.st_Menu_Text hinterlegt. Dabei werden die einzelnen Element mittels '#' voneinander getrennt. Um einzelne Sub-Menu Elemente voneinander abzugrenzen bzw. mit einer Trennlinie zu versehen, muss als Text des Menu-Elementes ein '-' angegeben werden.

Innerhalb des Sub-Menu kann mit Cursor oben/unten navigiert werden. Wird ein Sub-Menu-Element mit Enter/Return Taste bestätigt, so wird bei *.in_Menu_Selected die Nummer des gewählten Menu-Punktes ausgegeben.

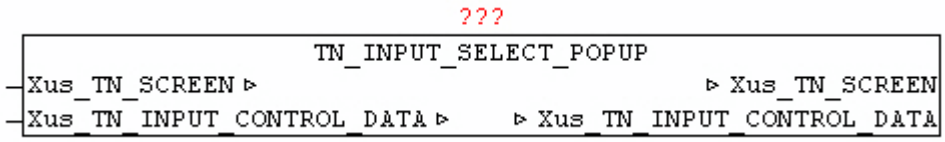
Ein aktives Menu-Popup sichert automatisch den Hintergrund bevor es gezeichnet wird, und restauriert den Hintergrund wieder nach Beendigung.

Solange eine Menu-Anzeige aktiv ist, dürfen vom Anwenderprogramm aber keinerlei grafisches Veränderungen gemacht werden. Dies kann mittels TN_SCREEN.bo_Menue_Bar_Dialog = TRUE bzw. TN_SCREEN.bo_Modal_Dialog = TRUE überprüft werden.

Der Baustein wird primär vom TN_INPUT_MENU_BAR und TN_INPUT_SELECT_POPUP intern benutzt, und muss nicht direkt vom Anwender ausgeführt werden.

11.7. TN_INPUT_SELECT_POPUP

Type	Funktionsbaustein
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN
	Xus_TN_INPUT_CONTROL : us_TN_INPUT_CONTROL



Der Baustein TN_INPUT_SELECT_POPUP dient zum verwalten einer Auswahl von Texten, durch anzeige eines Popup-Dialogs. Dazu muss *.in_Type = 3 gesetzt werden.

Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.

Die Auswahltexte werden über *.st_Input_Data übergeben. Die Text-Element müssen durch das Zeichen '#' von einander getrennt werden.

Besitzt das Element den Fokus , kann mittels der Enter/Return Taste der Auswahl-Dialog aktiviert werden.

Mittels Cursor Hoch/Runter kann zwischen den einzelnem Elementen gewechselt werden. Wird der Anfang bzw. das Ende der Liste erreicht so wird an der gegenüberliegenden Seite fortgesetzt

Die Text-Element wird dabei mittels *.st_Input_Mask verknüpft, das heißt damit kann die Ausgabe-Textlänge nachträglich beeinflusst werden.

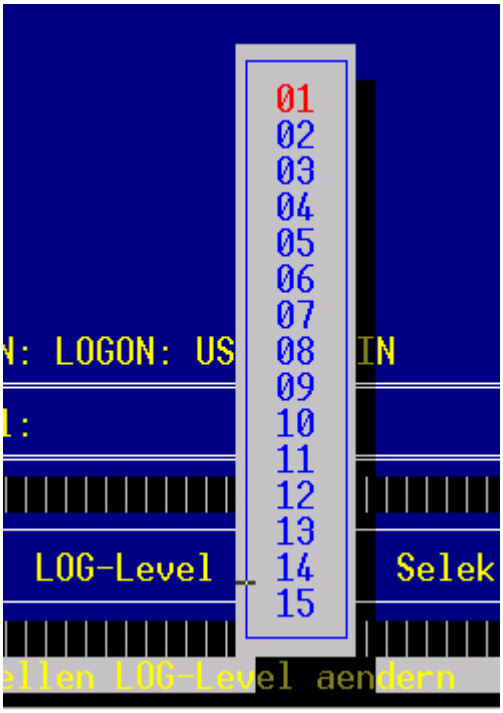
Durch Betätigen der Eingabe/Return Taste wird der Text des gewählten Elementes bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden.

Eine aktive Auswahl (Auswahl-Dialog) kann jederzeit mit der Escape-Taste abgebrochen werden.

Beispiel:

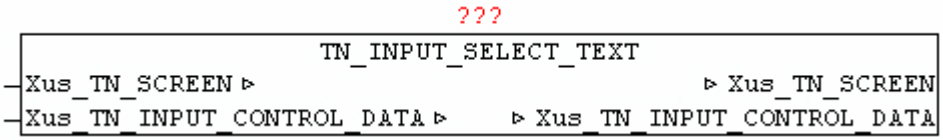
```
*.in_Type := 03;
*.in_Y := 20;
*.in_X := 18;
*.by_Attr_mF := 16#17;
*.by_Attr_oF := 16#47;
*.st_Input_ToolTip :=
            ' aktuellen LOG-Level aendern     | press enter to select |';
*.in_Input_Option := 00;
*.in_Title_Y_Offset := 00;
*.in_Title_X_Offset := 00;
*.by_Title_Attr := 16#34;
*.st_Title_String := ' LOG-Level ';
*.st_Input_Mask := '  ';
*.st_Input_Data :=
            '01#02#03#04#05#06#07#08#09#10#11#12#13#14#15';
```

Ergibt folgende Ausgabe:



11.8. TN_INPUT_SELECT_TEXT

Type	Funktionsbaustein
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN Xus_TN_INPUT_CONTROL : us_TN_INPUT_CONTROL



Der Baustein TN_INPUT_SELECT_TEXT dient zum verwalten einer Auswahl von Texten. Dazu muss *.in_Type = 2 gesetzt werden.

Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.

Die Auswahltexte werden über *.st_Input_Data übergeben. Die Text-Element müssen durch das Zeichen '#' von einander getrennt werden.

Besitzt das Element den Fokus , kann mittels der Leertaste (Space) zwischen den einzelnen Texten gewechselt werden. Das ausgewählte Text-

Element wird dabei mittels *.st_Input_Mask verknüpft, das heißt damit kann die Ausgabe-Textlänge nachträglich beeinflusst werden.

Durch Betätigen der Eingabe/Return Taste wird der Eingabetext bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden.

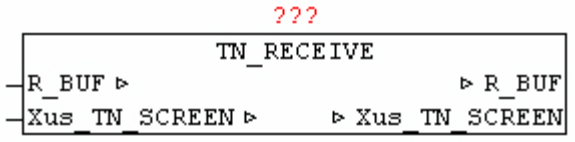
Beispiel:

```
*.in_Type := 2;
*.in_Y := 20;
*.in_X := 58;
*.by_Attr_mF := 16#17;
*.by_Attr_oF := 16#47;
*.st_Input_ToolTip := ' Selektion-Text aktiv      | press space to select |';
*.in_Input_Option := 02;
*.in_Title_Y_Offset := 00;
*.in_Title_X_Offset := 00;
*.by_Title_Attr := 16#34;
*.st_Title_String := ' Betriebsart ';
*.st_Input_Mask := '              ';
*.st_Input_Data := '<Auto>#<Hand>#<Stop>#<Restart>';
```

Ergibt folgende Ausgabe:

11.9. TN_RECEIVE

Type	Funktionsbaustein
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN
	R_BUF : NETWORK_BUFFER (Telnet Empfangsbuffer)



Der Baustein TN_RECEIVE empfängt die Eingabedaten vom Telnet-Client und wertet die Tastencodes aus.

Liegt der Tastencode im Bereich 32-126 so wird dieser als ASCII-Code unter Xus_TN_SCREEN.by_Input_ASCII_Code abgelegt. Zusätzlich wird Xus_TN_SCREEN.bo_Input_ASCII_IsNum = TRUE wenn dieser eine Ziffer zwischen 0 und 9 entspricht.

Ist der Tastencode einer der folgenden Extended-Codes dann wird dieser unter Xus_TN_SCREEN.by_Input_Exten_Code abgelegt.

Exten_code	**Tastenbezeichnung**
65	Cursor oben
66	Cursor unten
67	Cursor rechts
68	Cursor links
72	Pos1
75	Ende
80	F1
81	F2
82	F3
83	F4
8	Backspace
9	Tabulator
13	Return (Enter)
27	Escape

11.10. TN_SEND_ROWS

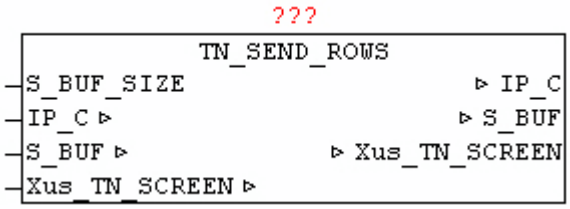
Type Funktionsbaustein

INPUT S_BUF_SIZE : UINT (Anzahl Bytes in S_BUF.BUFFER)

IN_OUT IP_C : IP_CONTROL (Verbindungsdaten)

S_BUF: NETWORK_BUFFER (Sendedaten)

Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SEND_ROWS dient zum automatischen Updaten der grafischen Änderungen am Telnet-Screen, indem die veränderten Zeilen an den Telnet-Client versendet werden.

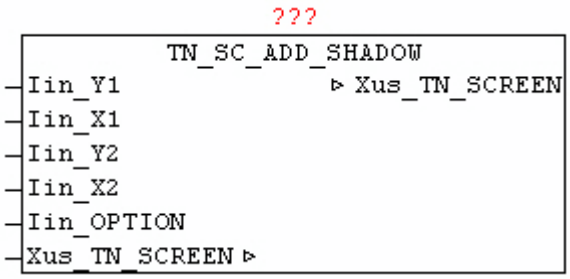
Wird am Telnet-Screen eine Farbe oder ein Zeichen in einer Zeile verändert, so wird immer automatisch diese Zeile zum Update markiert. Der Baustein überprüft ob bei Xus_TN_SCREEN.bya_Line_Update[0..23] eine oder mehrere Zeilen markiert sind, und erzeugt daraus eine ANSI-Code Byte-Stream der an den Telnet-Client gesendet wird. Weiters wird bei Xus_TN_SCREEN.bo_Clear_Screen = TRUE ein Clear-Screen ausgelöst. Bei erkennen einer neuen Telnet-Client Verbindung werden automatisch alle Zeilen zum Update markiert, damit der ganze Bildschirminhalt ausgegeben wird. Ist die erforderliche Datenmenge größer als S_BUF.BUFFER werden die Daten automatisch Blockweise ausgegeben.

11.11. TN_SC_ADD_SHADOW

Type Funktionsbaustein

INPUT: lin_Y1 : INT : (Y1-Koordinate der Fläche)
lin_X1 : INT : (X1-Koordinate der Fläche)
lin_Y2 : INT : (Y2-Koordinate der Fläche)
lin_X2 : INT : (X2-Koordinate der Fläche)
lin_OPTION : INT : (Art des Schattens)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN

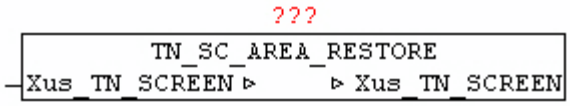


Der Baustein TN_SC_ADD_SHADOW ermöglicht das hinzufügen von optischen Schatten zu Rechteckigen Zeichenelementen. Durch Angabe einer rechteckigen Fläche mittels der Parameter X1,Y1 und X2,Y2 wird der Grundrahmen definiert, zu dem rechts und unten farblich abgedunkelter Linien gezeichnet werden. Die Schattenkoordinaten X1,Y1 und X2,Y2 werden immer +1 zum eigentlichen Grundelement angegeben. Mittels OPTION kann zwischen zwei Schattenvarianten gewählt werden. Wenn OPITION = 0 dann wird der Schatten durch reine Farbanpassung (Abdunkelung der Zeichen) erreicht. Wird eine OPTION > 0 angegeben, werden im Bereich des Schattens alle Zeichen durch schwarze ausgefüllte Zeichen ersetzt.

11.12. TN_SC_AREA_RESTORE

Type Funktionsbaustein

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



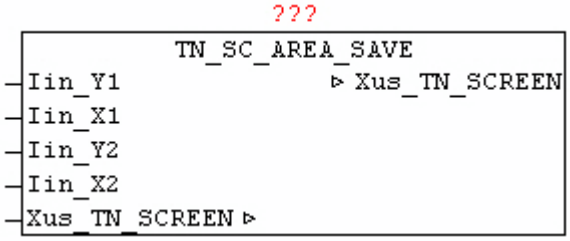
Der Baustein TN_SC_AREA_RESTORE ermöglicht das wiederherstellen des zuvor gesicherten Bildschirmbereiches.Die im Datenbereich Xus_TN_SCREEN.bya_BACKUP[x] abgelegten Bildschirmdaten werden mit Hilfe der hinterlegten Koordinaten wiederhergestellt. Dies wird vorwiegend nach dem Aufruf vom Baustein MENU-BAR und MENU-POPUP durchgeführt, um den veränderten Bildschirmbereich wiederherzustellen.

11.13. TN_SC_AREA_SAVE

Type Funktionsbaustein

INPUT: lin_Y1 : INT : (Y1-Koordinate der Fläche)
lin_X1 : INT : (X1-Koordinate der Fläche)
lin_Y2 : INT : (Y2-Koordinate der Fläche)
lin_X2 : INT : (X2-Koordinate der Fläche)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



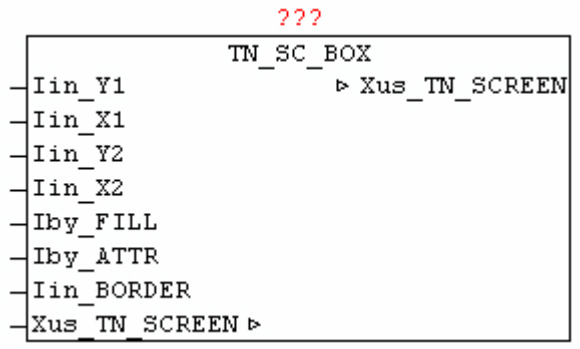
Der Baustein TN_SC_AREA_SAVE ermöglicht das sichern von rechteckigen Flächen des Bildschirms bevor dieser durch anderen Zeichenoperationen verändert wird. Dies wird vorwiegend vor dem Aufruf vom Baustein MENU-BAR und MENU-POPUP gemacht , da diese die Elemente als Overlay-Grafik darstellen. Mittels X1,Y1 und X2,Y2 werden die Koordinaten des zu sichernden Bildschirmbereichs angegeben. Dabei werden die Daten in den Datenbereich Xus_TN_SCREEN.bya_BACKUP[x] gesichert. Es werden hierbei die Koordinaten und die eigentlichen Zeichen und Farbinformationen darin abgelegt. Der Buffer kann maximal die halbe Fläche des Bildschirms aufnehmen.

11.14. TN_SC_BOX

Type Funktionsbaustein

INPUT: Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1 : INT : (X1-Koordinate der Fläche)
Iin_Y2 : INT : (Y2-Koordinate der Fläche)
Iin_X2 : INT : (X2-Koordinate der Fläche)
Iby_FILL : BYTE : (Charakter zum füllen der Fläche)
Iby_ATTR : BYTE : (Farbcode zum füllen der Fläche)
Iby_BORDER : BYTE : (Typ der Umrahmung)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_BOX dient zum zeichnen eines rechteckigen Bereiches,der mit dem bei Iby_FILL angegebenen Charakter gefüllt wird. Mit Parameter Iby_ATTR kann die Füllfarbe vorgegeben werden. Der Füllbereich wird mit einer Umrandung gezeichnet, die mittels Iin_BORDER vorgegeben wird.

Border-Typen:

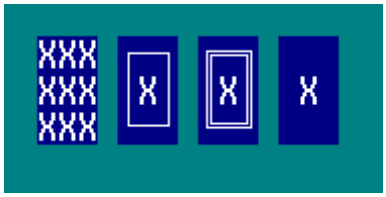
0 = kein Rahmen

1 = Rahmen mit Einzellinie

2 = Rahmen mit Doppelllinie

3 = Rahmen mit Leerzeichen

Beispiel: Box mit Füllzeichen 'X' und Farbe Weiß auf Blau



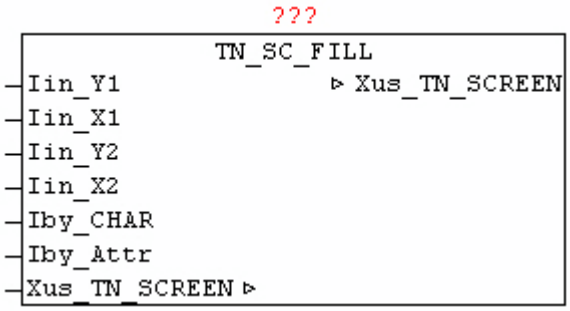
Darstellung mit Iin_BORDER Wert 0,1,2 und 3 (von Links nach Rechts)

11.15. TN_SC_FILL

Type Funktionsbaustein

INPUT: Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1 : INT : (X1-Koordinate der Fläche)
Iin_Y2 : INT : (Y2-Koordinate der Fläche)
Iin_X2 : INT : (X2-Koordinate der Fläche)
Iby_CHAR : BYTE : (Charakter zum füllen der Fläche)
Iby_ATTR : BYTE : (Farbcode zum füllen der Fläche)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_FILL dient zum zeichnen eines rechteckigen Bereiches, der mit dem bei Iby_FILL angegebenen Charakter gefüllt wird.

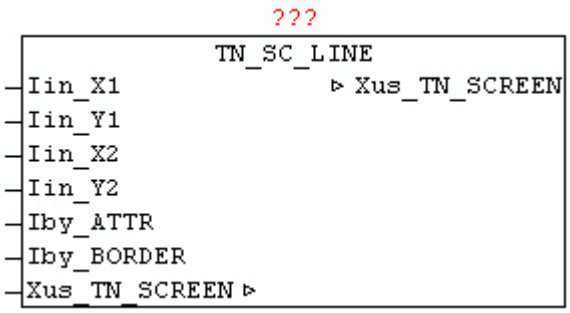
Beispiel: Box mit Füllzeichen 'X' und Farbe Weiß auf Blau

11.16. TN_SC_LINE

Type Funktionsbaustein

INPUT: Iin_Y1 : INT : (Y1-Koordinate der Linie)
Iin_X1 : INT : (X1-Koordinate der Linie)
Iin_Y2 : INT : (Y2-Koordinate der Linie)
Iin_X2 : INT : (X2-Koordinate der Linie)
Iby_ATTR : BYTE : (Farbcode der Linie)
Iby_BORDER : BYTE : (Typ der Linie)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_LINE dient zum zeichnen von waagrechten und senkrechten Linien. Mittels der X1/Y1 und X2/Y2 Koordinaten wird der Beginn und das Ende der Linie definiert. Die Linien-Type wird mittels Iin_BORDER und der Farbcode mit Iby_ATTR übergeben. Wird beim Linien zeichnen eine andere Linie dieser Type geschnitten, so wird automatisch das passende Kreuzungszeichen benutzt.

Border-Typen:

1 = Linie mit Einzellinie

2 = Linie mit Doppelllinie

>2 = Linie wird mit dem bei Iin_BORDER angegebenen Charakter gezeichnet

Beispiel:

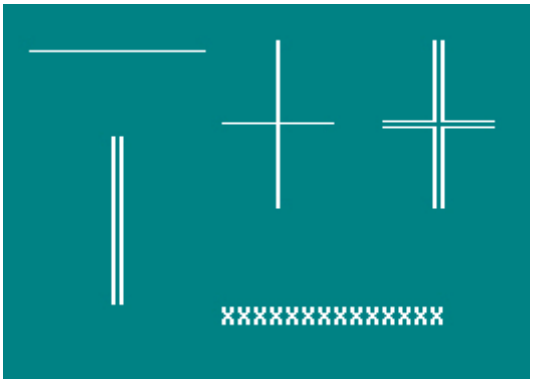
Horizontale Linie: Typ Single-Line

Vertikale Linie: Typ Double-Line

Horizontale und Vertikale Linie gekreuzt: Typ Single-Line

Horizontale und Vertikale Linie gekreuzt: Typ Double-Line

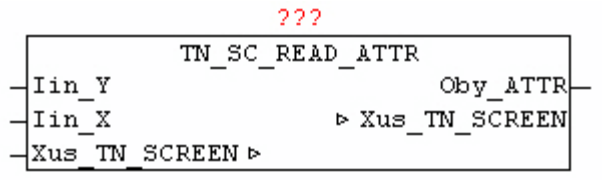
Horizontale Linie: Typ Charakter (X)



11.17. TN_SC_READ_ATTR

Type Funktionsbaustein

INPUT	Iin_Y : INT : (Y-Koordinate)
	Iin_X : INT : (X-Koordinate)
OUTPUT	Oby_ATTR : BYTE : (Farbinformation an Position X/Y)
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN

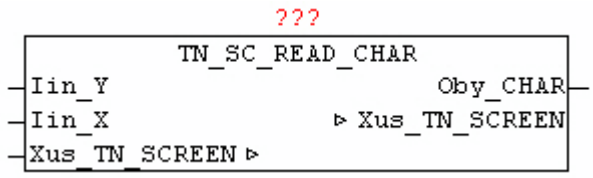


Der Baustein TN_SC_READ_ATTR dient zum auslesen der aktuellen Farbe des Charakters an der angegebenen Position X/Y.

11.18. TN_SC_READ_CHAR

Type	Funktionsbaustein

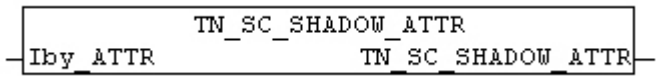
INPUT	Iin_Y : INT : (Y-Koordinate)
	Iin_X : INT : (X-Koordinate)
OUTPUT	Oby_CHAR : BYTE : (Zeichen an Position X/Y)
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_READ_CHAR dient zum auslesen des aktuellen Zeichen an der angegebenen Position X/Y.

11.19. TN_SC_SHADOW_ATTR

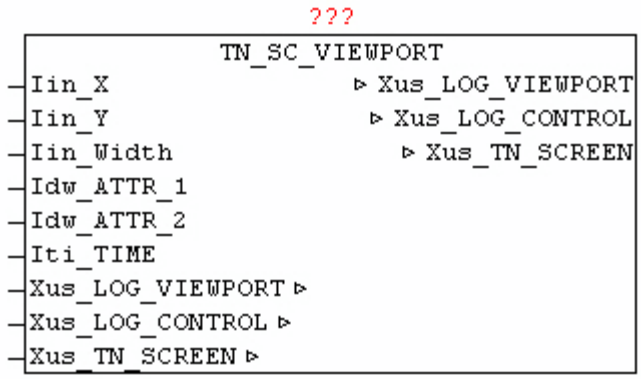
Type	Funktion : BYTE
INPUT	Iby_ATTR : BYTE : (Farbinformation)



Der Baustein TN_SC_SHADOW_ATTR konvertiert eine helle Farbe zu einer dunklen Farbe.

11.20. TN_SC_VIEWPORT

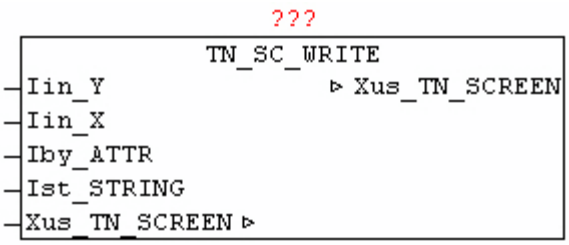
Type	Funktionsbaustein
INPUT	Iin_Y : INT : (Y-Koordinate)
	Iin_X : INT : (X-Koordinate)
	Iin_Width : INT : (Breite des Fenster - Anzahl der Zeichen)
	Idw_ATTR_1 : DWORD : (Farbe 1,2,3 und 4)
	Idw_ATTR_2 : DWORD : (Farbe 5,6,7 und 8)
	Iti_TIME : TIME : (Aktualisierungszeit)
IN_OUT	Xus_LOG_VIEWPORT : LOG_VIEWPORT
	Xus_LOG_CONTROL : LOG_CONTROL
	Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_VIEWPORT dient zum anzeigen der Meldungen von der Datenstruktur LOG_CONTROL innerhalb eines rechteckigen Bereiches am Bildschirm. Die anzuzeigenden Meldungen werden zuvor mit Hilfe des Baustein LOG_VIEWPORT aufbereitet, und wenn erforderlich über Xus_LOG_VIEWPORT.UPDATE eine Aktualisierung ausgelöst. Mittels Iin_X und Iin_Y wird die linke obere Ecke des Sichtfenster, und mit Iin_Width die breite des Sichtfensters definiert. Die Anzahl der darzustellenden Zeilen wird durch Xus_LOG_VIEWPORT.COUNT vorgegeben. Die hinterlegte Farbinformation in Xus_LOG_CONTROL.MSG_OPTION[x] pro Meldung wird auf die parametrierten Farbcodes bei Idw_ATTR_1 und Idw_ATTR2 automatisch konvertiert, somit können die Farben bei der Darstellung immer individuell angepasst werden. Die Meldungen werden immer automatisch auf die breite des Sichtfenster gekürzt bzw. Abgeschnitten.

11.21. TN_SC_WRITE

Type	Funktionsbaustein
INPUT	Iin_Y : INT (Y-Koordinate)
	Iin_X : INT (X-Koordinate)
	Iby_ATTR : BYTE : (Farbcode - Schriftfarbe)
	Ist_STRING : STRING (Text)
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN

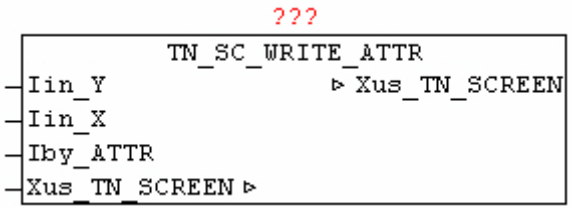


Der Baustein TN_SC_WRITE gibt an der angegebenen Koordinate Iin_Y, Iin_Y den Text Ist_STRING mit der Farbe Iby_ATTR aus.

Wird als Farbcode = 0 angegeben, dann wird der String ausgegeben, ohne das die vorhandenen alten Farbinformationen an den jeweiligen Zeichenpositionen verändert werden.

11.22. TN_SC_WRITE_ATTR

Type	Funktionsbaustein
INPUT	Iin_Y : INT (Y-Koordinate)
	Iin_X : INT (X-Koordinate)
	Iby_ATTR : BYTE : (Farbcode)
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_WRITE_ATTR ändert an der angegebenen Koordinate Iin_Y, Iin_Y den Farbcode ohne das vorhandene Zeichen an der Position zu verändern.

11.23. TN_SC_WRITE_C

Type	Funktionsbaustein
INPUT	Iin_Y : INT (Y-Koordinate)
	Iin_X : INT (X-Koordinate)

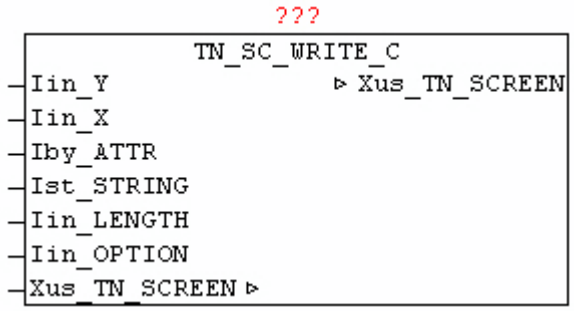
Iby_ATTR : BYTE : (Farbcode)

Ist_STRING : STRING : (Text)

Iin_LENGTH : INT : (Text wird auf diese Länge angepasst)

Iin_OPTION : INT : (Option der Text-Längenanpassung)

IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_WRITE_C gibt an der angegebenen Koordinate Iin_Y, Iin_Y den Text Ist_STRING mit der Farbe Iby_ATTR aus. Der Text wird vor der Ausgabe auf die Länge Iin_LENGTH angepasst, und mittels Iin_OPTION kann die Textposition bestimmt werden.

Iin_OPTION

0 = rechts mit Leerzeichen auffüllen z.B. 'TEST '

1 = links mit Leerzeichen auffüllen z.B. ' TEST'

2 = zentrieren und mit Leerzeichen auffüllen z.B. ' TEST '

11.24. TN_SC_WRITE_CHAR

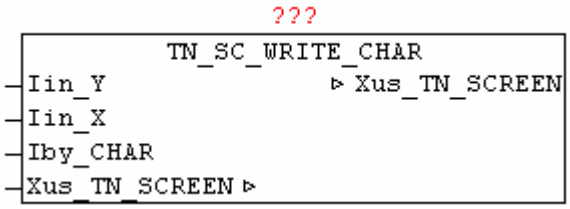
Type Funktionsbaustein

INPUT Iin_Y : INT : (Y-Koordinate)

Iin_X : INT : (X-Koordinate)

OUTPUT Iby_CHAR : BYTE : (Zeichen)

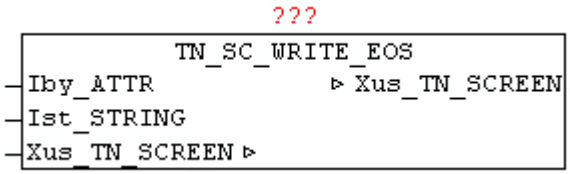
IN_OUT Xus_TN_SCREEN : us_TN_SCREEN



Der Baustein TN_SC_WRITE_CHAR gibt das Zeichen Iby_CHAR an der angegebenen Koordinate Iin_Y, Iin_X aus, und verändert dabei die Farbinformation an der angegebene Position nicht.

11.25. TN_SC_WRITE_EOS

Type	Funktionsbaustein
INPUT	Iby_ATTR : BYTE : (Farbcode - Schriftfarbe)
	Ist_STRING : STRING (Text)
IN_OUT	Xus_TN_SCREEN : us_TN_SCREEN



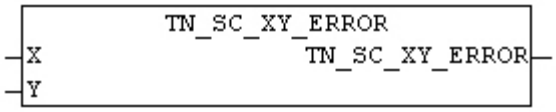
Der Baustein TN_SC_WRITE_EOS gibt an der Endposition des zuletzt mit TN_SC_WRITE oder TN_SC_WRITE_EOS ausgegebenen Textes den Text Ist_STRING mit der Farbe Iby_ATTR aus.

Damit können fortlaufend Texte ausgegeben werden, ohne das immer extra die neuen Koordinaten mit angegeben werden müssen.

11.26. TN_SC_XY_ERROR

Type	Funktion : BOOL

INPUT: X : INT : (X-Koordinate)
Y : INT : (Y-Koordinate)

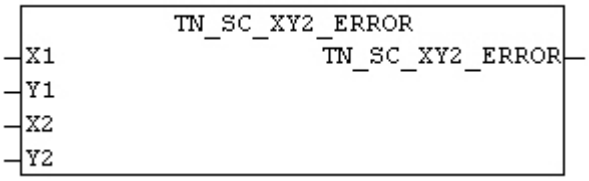


Der Baustein TN_SC_XY_ERROR prüft ob sich die angegebene Koordinate innerhalb des Bildschirmbereiches befindet. Wenn die Überprüfung negativ ausfällt wird als Ergebnis TRUE ausgegeben.

11.27. TN_SC_XY2_ERROR

Type Funktion : BOOL

INPUT: X1 : INT : (X1-Koordinate der Fläche)
Y1 : INT : (Y1-Koordinate der Fläche)
X2 : INT : (X2-Koordinate der Fläche)
Y2 : INT : (Y2-Koordinate der Fläche)



Der Baustein TN_SC_XY2_ERROR prüft ob sich die angegeben Koordinaten innerhalb des Bildschirmbereiches befindet. Die Fläche darf nicht über den Bildschirmrand hinaus gehen. Wenn die Überprüfung negativ ausfällt wird als Ergebnis TRUE ausgegeben.

12. Netzwerk-Variablen

12.1. Konzept

Das Bausteinpaket NET_VAR_* ermöglicht den bidirektionale Prozessdatenaustausch zwischen zwei Steuerungen auf der die network.lib nutzbar ist. Dabei wird zwischen den beiden Steuerungen eine Punkt zu Punkt (P2P) Verbindung aufgebaut. Die Prozessdaten können mittels der Bausteine

NET_VAR_BOOL8
NET_VAR_DWORD
NET_VAR_BUFFER
NET_VAR_STRING
NET_VAR_REAL

erfasst bzw. ausgegeben werden. Jeder dieser Bausteine besitzt Eingangsprozessdaten und Ausgangsprozessdaten die automatisch mit der Gegenseite (anderer Steuerung) ausgetauscht werden.

IN-Daten auf der einen Seite werden als OUT Daten auf der Gegenseite wieder ausgegeben.

Damit können auf einfache Weise zwischen gleichen Steuerungen aber auch zwischen verschiedenen Steuerungen und Plattformen (WAGO,BECKHOFF,PHOENIC CONTACT) Prozessdaten ausgetauscht werden.

Vorgangsweise zur Erstellung der des Master-Bausteins:

Zuerst werden alle benötigten Prozessdaten mittels der NET_VAR_* Bausteine Instanzen parametriert bzw. übergeben. Abschließend muss einmal der NET_VAR_CONTROL ausgerufen werden, der die Prozessdaten dann mit der Gegenseite automatisch austauscht. Die IP-Adresse der Gegenseite und MASTER = TRUE muss vorgegeben werden.

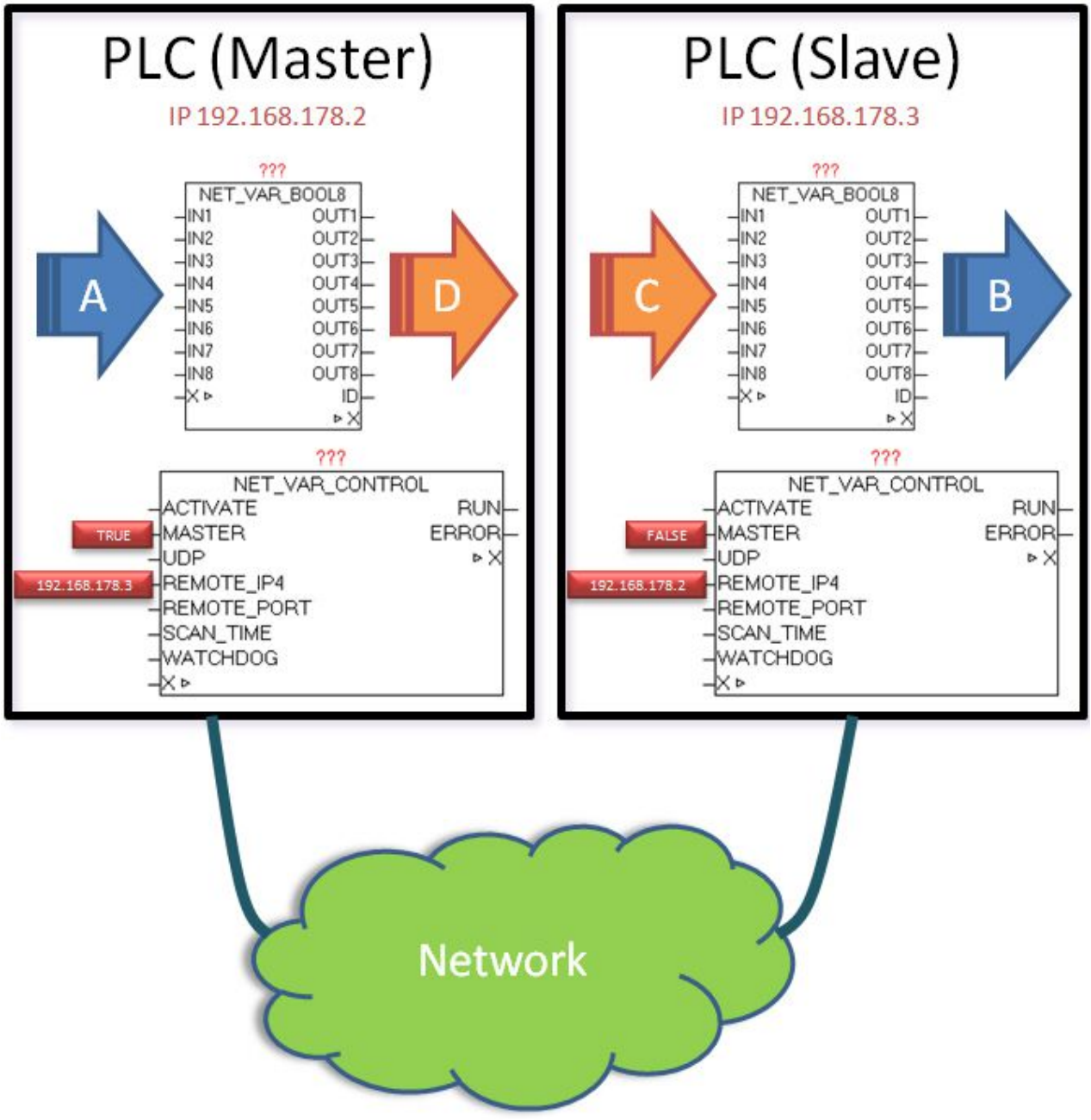
Vorgangsweise zur Erstellung der des SLAVE-Bausteins:

Der zuvor erstellte Master-Baustein muss einfach 1:1 kopiert werden. Die IP-Adresse muss durch die der Gegenseite ersetzt , und bei MASTER=FALSE vorgegeben werden.

Beispiel: (Siehe nachfolgendes Schema)

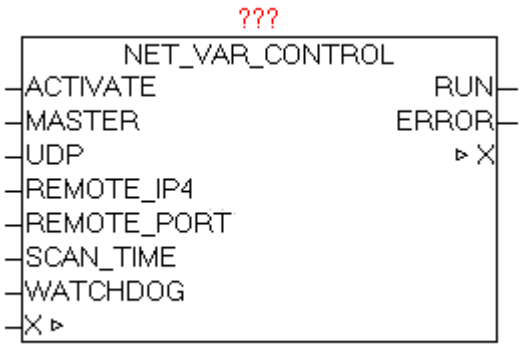
Die Eingangsdaten (A) vom PLC-Master werden beim Baustein NET_VAR_BOOL8 übergeben und mittels NET_VAR_CONTROL zur anderen Steuerung (PLC-SLAVE) übertragen,und dort wieder beim gleichen NET_VAR_BOOL8 Baustein bei den Ausgangsdaten (B) wieder ausgegeben.

Die Eingangsdaten (C) vom PLC-Slave werden beim Baustein NET_VAR_BOOL8 übergeben und mittels NET_VAR_CONTROL zur anderen Steuerung (PLC-Master) übertragen,und dort wieder beim gleichen NET_VAR_BOOL8 Baustein bei den Ausgangsdaten (D) wieder ausgegeben.



12.2. NET_VAR_CONTROL

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT	ACTIVATE : BOOL (Aktiviert den Datenaustausch)
	MASTER : BOOL (FALSE=SLAVE / TRUE = MASTER)
	UDP : BOOL (FALSE=TCP / TRUE = UDP)
	REMOTE_IP4 : DWORD (IP4-Adresse der anderen SPS)
	REMOTE_PORT : WORD (PORT-Nummer der anderen SPS)
	SCAN_TIME : TIME (Aktualisierungszeit)
	WATCHDOG : TIME (Überwachungszeit)
OUTPUT	RUN : BOOL (Datenaustausch aktiv – kein Fehler)
	ERROR : DWORD ((Fehlercode)



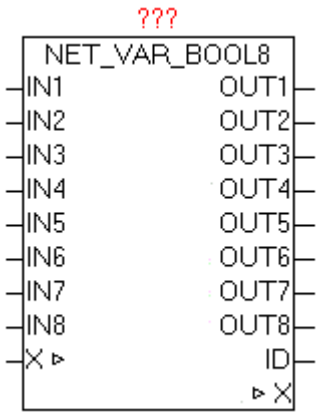
Der Baustein NET_VAR_CONTROL koordiniert den Datenaustausch zwischen den beiden Steuerungen und den Satellitenbausteinen NET_VAR_*. Mit ACTIVATE = TRUE kann der Datenaustausch freigegeben werden. Der Baustein muss auf beiden Steuerungen aufgerufen werden, wobei der Parameter MASTER einmal mit TRUE und einmal mit FALSE belegt werden muss. Damit wird bestimmt welche Seite den aktiven Verbindungsaufbau vornimmt. Mit UDP (FALSE/TRUE) kann vorgeben werden, ob eine UDP oder TCP-Verbindung benutzt wird. Die IP-Adresse der Gegenseite muss bei REMOTE-IP4 vorgeben werden, und alternativ auch die Port-Adresse (Standard-Port ist 10000). Die SCAN-TIME bestimmt in welchen Zeitabstand eine Datenaktualisierung durchgeführt wird (Standardwert ist T#1s). Über WATCHDOG wird die Überwachungszeit vorgegeben (Standardwert ist T#2s).Bei funktionierenden Datenaustausch wird Parameter RUN = TRUE. Ist der Datenaustausch länger als die Watchdog Zeit nicht möglich , wird RUN = FALSE und ein Fehler ausgegeben. Der Fehler muss nicht quittiert werden, da der Baustein selbstständig versucht den Datenaustausch wiederherzustellen. Sobald kein Fehler mehr vorhanden ist, wird RUN = TRUE und der Fehlercode wird wieder gelöscht.

ERROR: (als HEX-Wert betrachten !)

DWORD				**Meldungstype**	Beschreibung
B3	B2	B1	B0		
XX	..	..	..	Verbindungsaufbau	Connect-Error - Siehe Baustein IP_CONTROL
..	**XX**	..	..	Daten senden	Sende-Error - Siehe Baustein IP_CONTROL
..	..	**XX**	..	Daten empfangen	Empfangs-Error - Siehe Baustein IP_CONTROL
..	..	..	**XX**	Konfigurationsfehler	ID-Nummer des Baustein

12.3. NET_VAR_BOOL8

Type Funktionsbaustein:
IN_OUT X : NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT IN1..8 : BOOL (Signaleingänge)
OUTPUT OUT1..8 : BOOL (Signalausgänge)
ID : BYTE (Identifikationsnummer)



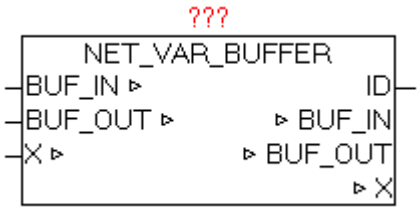
Der Baustein NET_VAR_BOOL8 dient zum bidirektionalen Übertragen von 8 binären Signalen vom Master zum Slave und umgekehrt. Die Signale IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.

Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangsdaten hier als OUT1..8 wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

12.4. NET_VAR_BUFFER

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
	BUF_IN : ARRAY[1..64] OF BYTE (Eingang-Datenpuffer)
	BUF_OUT : ARRAY[1..64] OF BYTE (Ausgang-Datenpuffer)
OUTPUT	ID : BYTE (Identifikationsnummer)



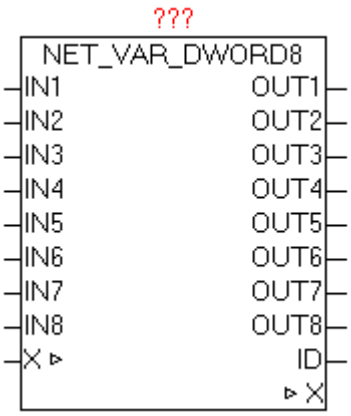
Der Baustein NET_VAR_BUFFER dient zum bidirektionalen Übertragen von 64 Bytes vom Master zum Slave und umgekehrt. Die Daten von BUF_IN werden erfasst und an der anderen Seite (andere Steuerung) am gleichen Baustein an der gleichen Position als BUF_OUT wieder ausgegeben.

Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangsdaten hier als BUF_OUT wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

12.5. NET_VAR_DWORD8

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT	IN1..8 : DWORD (Eingangs DWORD)
OUTPUT	OUT1..8 : DWORD (Ausgangs DWORD)
	ID : BYTE (Identifikationsnummer)



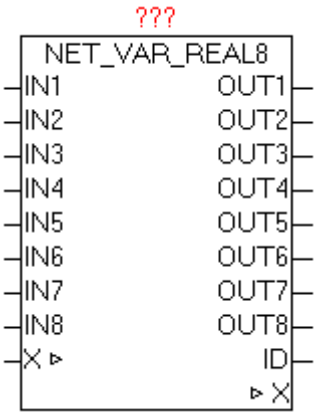
Der Baustein NET_VAR_DWORD8 dient zum bidirektionalen Übertragen von acht DWORD vom Master zum Slave und umgekehrt. Die DWORD IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.

Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangs-Datenwords hier als OUT1..8 wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

12.6. NET_VAR_REAL8

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT	IN1..8 : REAL (Eingangwert)
OUTPUT	OUT1..8 : REAL (Ausgangswert)
	ID : BYTE (Identifikationsnummer)



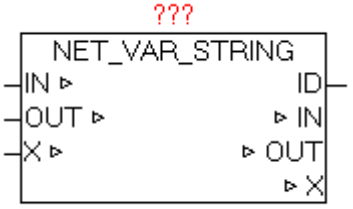
Der Baustein NET_VAR_REAL8 dient zum bidirektionalen Übertragen von acht REAL-Werten vom Master zum Slave und umgekehrt. Die REAL-Werte IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.

Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangs-REAL-Werte hier als OUT1..8 wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

12.7. NET_VAR_STRING

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
	IN : STRING(STRING_LENGTH) (Eingangs-String)
	OUT : STRING(STRING_LENGTH) (Ausgangs-String)
OUTPUT	ID : BYTE (Identifikationsnummer)



Der Baustein NET_VAR_STRING dient zum bidirektionalen Übertragen von einem STRING vom Master zum Slave und umgekehrt. Der STRING bei Parameter IN wird

erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT Parameter wieder ausgegeben.

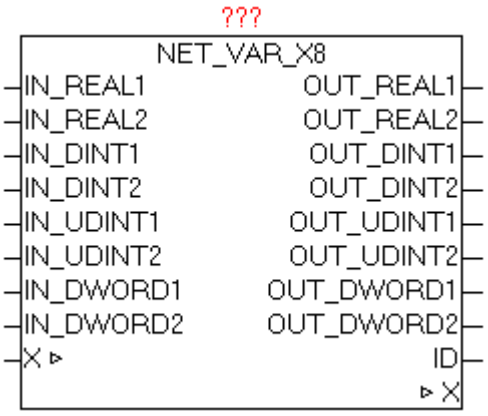
Gleichzeitig wird der an der Gegenseite (andere Steuerung) übergebenen Eingangs-STRING-Wert hier als OUT wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

12.8. NET_VAR_X8

Type	Funktionsbaustein:
IN_OUT	X : NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT	IN_REAL1 : REAL (Eingangwert)
	IN_REAL2 : REAL (Eingangwert)
	IN_DINT1 : DINT (Eingangwert)
	IN_DINT2 : DINT (Eingangwert)
	IN_UDINT1 : DINT (Eingangwert)
	IN_UDINT2 : DINT (Eingangwert)
	IN_DWORD1 : DINT (Eingangwert)
	IN_DWORD2 : DINT (Eingangwert)
OUTPUT	OUT_REAL1 : REAL (Ausgangwert)
	OUT_REAL2 : REAL (Ausgangwert)
	OUT_DINT1 : DINT (Ausgangwert)
	OUT_DINT2 : DINT (Ausgangwert)
	OUT_UDINT1 : DINT (Ausgangwert)
	OUT_UDINT2 : DINT (Ausgangwert)
	OUT_DWORD1 : DINT (Ausgangwert)
	OUT_DWORD2 : DINT (Ausgangwert)
	ID : BYTE (Identifikationsnummer)

Der Baustein NET_VAR_X8 dient zum bidirektionalen Übertragen von je zwei REAL,DINT,UDINT,DWORD Werten vom Master zum Slave und umgekehrt. Die



Eingangswerte werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position wieder ausgegeben.

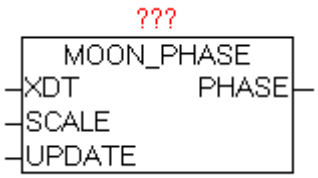
Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangswerte hier wieder ausgegeben.

Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.

13. Wetter-Daten

13.1. MOON_PHASE

Type	Funktionsbaustein:
INPUT	XDT : DT (Datum / Uhrzeit)
	SCALE : BYTE (Skalierungsfaktor)
	UPDATE : TIME (Aktualisierungszeit)
OUTPUT	PHASE : BYTE (Skalierte Wert der Mondphase)



Der Baustein MOON_PHASE dient berechnen der Mondphase zum angegeben Datum. Am Parameter XDT wird das aktuelle Datum und die Zeit übergeben, und immer nach Ablauf der Zeit bei Parameter „UPDATE“ neu berechnet. Der Standardwert für UPDATE ist 1 Stunde, und für den Skalierungsfaktor 12.

Eine Mondphase dauert ca. 29.53 Tage, und durchläuft hierbei die typischen Zustände von Neumond bis Vollmond bzw. zunehmender und abnehmender Mond). Dieser Zyklus kann mittels SCALE auf einen gewünschten Wert zwischen 0 und 255 skaliert werden. Wird hier z.B, 100 angegeben, so wird die Mondphase als Prozentwert ausgegeben.

Die tatsächliche Länge einer einzelnen Mond-Periode, ist verhältnismäßig großen Schwankungen unterworfen, und wird von der verwendeten Berechnungsmethode nicht berücksichtigt. Somit ergeben sich mitunter Abweichungen von einigen Stunden. Auch der Betrachtungsort (Geo-Position) ist ein virtueller Punkt im Erdmittelpunkt.

Soll die Mondphase mittels Grafik visualisiert werden kann der Stand-Skalierungsfaktor von 12 benutzt werden, um damit die Stufen 0-11 zu erhalten

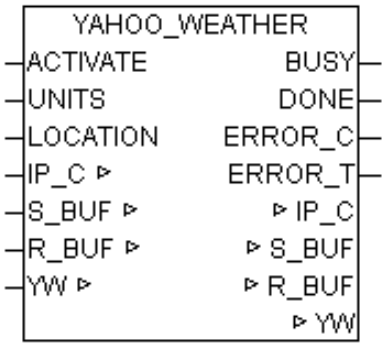
Siehe Kapitel Visualisierung - Mond-Grafiken

http://de.wikipedia.org/wiki/Mondphase

13.2. YAHOO_WEATHER

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
	YW: YAHOO_WEATHER (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Abfrage)
	UNITS: BOOL (FALSE = Celsius , TRUE = Fahrenheit)
	LOCATION: STRING(20) (Ortsangabe mittels LOCATION-ID)
OUTPUT	BUSY: BOOL (Abfrage ist aktiv)
	DONE: BOOL (Abfrage ohne Fehler beendet)
	ERROR_C: DWORD (Fehlercode)
	ERROR_T: BYTE (Fehlertype)

???



Der Baustein lädt die aktuellen Wetterdaten des angegebene Ortes mittels eines RSS feed (XML-Datenstruktur) von http://weather.yahooapis.com herunter, analysiert die XML-Daten und legt die wesentlichen Daten aufbereitet in der YAHOO_WEATHER Datenstruktur ab. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet und eine DNS Abfrage mit nachfolgender HTTP-GET durchgeführt. Nach erfolgreichen Empfang der Daten werden mittels XML_READER alle Elemente durchlaufen und wenn benötigt in der Datenstruktur in konvertierter Form abgelegt. Mit UNITS kann noch zwischen Fahrenheit und Celsius als Einheit gewählt werden. Durch Angabe der LOCATION_ID wird der genaue Ort des Wetters angegeben. Während die Abfrage aktiv ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.

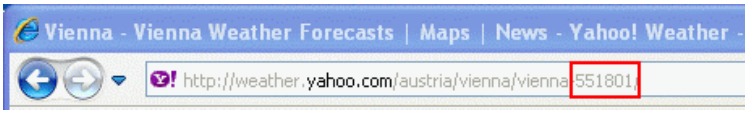
ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

Suchen der Location-ID eines bestimmten Ortes:

Mittels Internet-Browser die Seite http://weather.yahoo.com/ aufrufen, und im Eingabefeld “Enter city or zip code:” den Namen des gesuchten Ortes eingeben, und mittels “Go” suchen bzw. Aufrufen.

Nach erfolgreicher Auswahl werden im Browserfenster die aktuellen Wetterdaten des angegebenen Ortes angezeigt. In der URL (Web-Link) Zeile ist nun die Location-ID ersichtlich.



Somit ergibt der gesuchte Ort „Wien (vienna)“ die Location-ID „551801“. Dieser Code muss am Baustein als Parameter übergeben werden.

Beispiel eines RSS feed:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rss version="2.0" xmlns:yweather="http://weather.yahooapis.com/ns/rss/1.0"
 xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
```

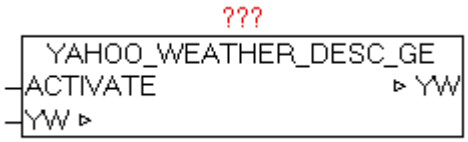
```
<channel>
  <title>Yahoo! Weather - Sunnyvale, CA</title>
  <link>http://us.rd.yahoo.com/dailynews/rss/weather/Sunnyvale__CA/
  *http://weather.yahoo.com/forecast/94089_f.html</link>
  <description>Yahoo! Weather for Sunnyvale, CA</description>
  <language>en-us</language>
  <lastBuildDate>Tue, 29 Nov 2005 3:56 pm PST</lastBuildDate>
  <ttl>60</ttl>
  <yweather:location city="Sunnyvale" region="CA" country="US"></yweather:location>
  <yweather:units temperature="F" distance="mi" pressure="in" speed="mph"></yweather:units>
  <yweather:wind chill="57" direction="350" speed="7"></yweather:wind>
  <yweather:atmosphere humidity="93" visibility="1609" pressure="30.12"  rising="0"></yweather:atmosphere>
  <yweather:astronomy sunrise="7:02 am" sunset="4:51 pm"></yweather:astronomy>
  <image>
    <title>Yahoo! Weather</title>
    <width>142</width>
    <height>18</height>
    <link>http://weather.yahoo.com/</link>
    <url>http://us.i1.yimg.com/us.yimg.com/i/us/nws/th/main_142b.gif</url>
  </image>
  <item>
    <title>Conditions for Sunnyvale, CA at 3:56 pm PST</title>
    <geo:lat>37.39</geo:lat>
    <geo:long>-122.03</geo:long>
    <link>http://us.rd.yahoo.com/dailynews/rss/weather/
    <span style="font-size: 0px"> </span>Sunnyvale__CA/*
    <span style="font-size: 0px"> </span>http://weather.yahoo.com/<span style="font-size: 0px"> </span>forecast/94089_f.html
    </link>
    <pubDate>Tue, 29 Nov 2005 3:56 pm PST</pubDate>
    <yweather:condition text="Mostly Cloudy" code="26" temp="57" date="Tue, 29 Nov 2005 3:56
      pm PST"></yweather:condition>
    <description><![CDATA[
 <img src="http://us.i1.yimg.com/us.yimg.com/i/us/we/52/26.gif" /><br />
 <b>Current Conditions:</b><br />
 Mostly Cloudy, 57 F<p />
 <b>Forecast:</b><BR />
  Tue - Mostly Cloudy. High: 62 Low: 45<br />
  Wed - Mostly Cloudy. High: 60 Low: 52<br />
  Thu - Rain. High: 61 Low: 46<br />
<br />
<a href="http://us.rd.yahoo.com/dailynews/rss/weather/Sunnyvale__CA/*http://weather.yahoo.com/forecast/94089_f.html">Full
Forecast at Yahoo! Weather</a><BR/>
 (provided by The Weather Channel)<br/>]]>
    </description>
```

```
    <yweather:forecast day="Tue" date="29 Nov 2005" low="45" high="62" text="Mostly Cloudy"
      code="27"></yweather:forecast>
    <yweather:forecast day="Wed" date="30 Nov 2005" low="52" high="60" text="Mostly Cloudy"
      code="28"></yweather:forecast>
    <guid isPermaLink="false">94089_2005_11_29_15_56_PST</guid>
  </item>
</channel>
</rss>
```

Aus den XML-Daten werden die benötigten Elemente aufbereitet und in der YAHOO_WEATHER Datenstruktur abgelegt.

13.3. YAHOO_WEATHER_DESC_DE

Type	Funktionsbaustein:
IN_OUT	YW: YAHOO_WEATHER_DATA (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Übersetzung)



Der Baustein ersetzt die originalen englischen Wetterbeschreibungen durch deutsche Texte. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Texte) in der YAHOO_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels YAHOO_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein YAHOO_WEATHER mit ACTIVATE verschalten werden.

Folgende Elemente werden angepasst:

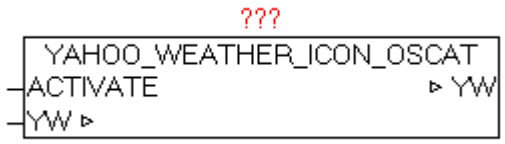
YW.CUR_CONDITIONS_TEXT

YW.FORCAST_TODAY_TEXT

YW.FORCAST_TOMORROW_TEXT

13.4. YAHOO_WEATHER_ICON_OSCAT

Type	Funktionsbaustein:
IN_OUT	YW: YAHOO_WEATHER_DATA (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Übersetzung)



Der Baustein ersetzt die originalen Anbieterspezifischen Wetter Icons-Nummern durch OSCAT-Standard Icon Nummern. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Icon-Nummern) in der YAHOO_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels YAHOO_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein YAHOO_WEATHER mit ACTIVATE verschalten werden.

Folgende Elemente werden angepasst:

YW.CUR_CONDITIONS_ICON

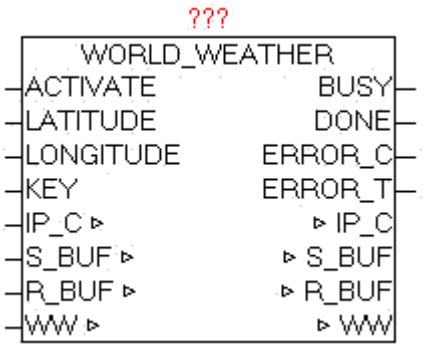
YW.FORCAST_TODAY_ICON

YW.FORCAST_TOMORROW_ICON

13.5. WORLD_WEATHER

Type	Funktionsbaustein:
IN_OUT	IP_C : IP_C (Parametrierungsdaten)
	S_BUF: NETWORK_BUFFER (Sendedaten)
	R_BUF: NETWORK_BUFFER (Empfangsdaten)
	WW: WORLD_WEATHER_DATA (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Abfrage)
	LATITUDE : REAL (Breitengrad des Bezugsortes)
	LONGITUDE : REAL (Längengrad des Bezugsortes)

	KEY: STRING(30) (API-Key)
OUTPUT	BUSY: BOOL (Abfrage ist aktiv)
	DONE: BOOL (Abfrage ohne Fehler beendet)
	ERROR_C: DWORD (Fehlercode)
	ERROR_T: BYTE (Fehlertype)



Der Baustein lädt die aktuellen Wetterdaten des angegebene Ortes von http://worldweather.com herunter, analysiert die Daten und legt die wesentlichen Daten aufbereitet in der WORLD_WEATHER_DATA Datenstruktur ab.

Vom aktuellen Tag werden folgende Werte abgelegt.

Observation time (UTC) , Temperature (°C) , Unique Weather Code , Weather description text, Wind speed in miles per hour , Wind speed in kilometre per hour , Wind direction in degree , 16-Point wind direction compass ,Precipitation amount in millimetre , Humidity (%) , Visibility (km) ,Atmospheric pressure in milibars , Cloud cover (%)

Vom aktuellen Tag und den darauffolgenden 4 Tagen werden folgende Werte abgelegt.

Date for which the weather is forecasted ,
Day and night temperature in °C(Celcius) and °F(Fahrenheit) ,
Wind Speed in mph (miles per hour) and kmph (Kilometer per hour) ,
16-Point compass wind direction , A unique weather condition code ,
Weather description text , Precipitation Amount (millimetre)

Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet und eine DNS Abfrage mit nachfolgender HTTP-GET durchgeführt. Nach erfolgreichen Empfang der Daten werden alle Elemente durchlaufen und wenn benötigt in der Datenstruktur in konvertierter Form abgelegt. Durch die Parameter LATITUDE und LONGITUDE wird der genaue Ort (Geografische Position) des Wetters angegeben. Während die Abfrage

aktiv ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.

ERROR_T:

Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

Neuen API-KEY anlegen:

Mittels Internet-Browser die Seite http://www.worldweatheronline.com aufrufen und mittels Button „Free sign up“ den Registrierungsdialog aufrufen, und die erforderlichen Felder ausfüllen. Nach der Registrierung wird eine Email versendet, die wiederum bestätigt werden muss, und in weiterer Folge wird in einer zweiten Email der persönliche API-Key versendet. Dieser API-Key muss beim Baustein am Parameter API-KEY übergeben werden.

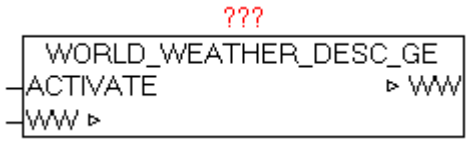
Breiten und Längengrad eines bestimmten Ortes bestimmen:

Mittels Internet-Browser die Seite http://www.mygeoposition.com/ aufrufen, und im Eingabefeld den Namen des gesuchten Ortes eingeben, und mittels “Geodaten berechnen” den Ort suchen. Danach wird der gesuchte Ort auf der Karte angezeigt inklusive des benötigten Breiten und

Längengrades in dezimaler Schreibweise. Die ermittelte Position gehört bei den Baustein-Parametern LATITUDE und LONGITUDE übergeben.

13.6. WORLD_WEATHER_DESC_DE

Type	Funktionsbaustein:
IN_OUT	WW: WORLD_WEATHER_DATA (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Übersetzung)



Der Baustein ersetzt die originalen englischen Wetterbeschreibungen durch deutsche Texte. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Texte) in der WORLD_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels WORLD_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein WORLD_WEATHER mit ACTIVATE verschalten werden.

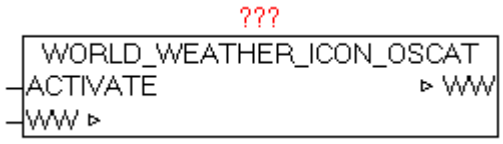
Folgende Elemente werden angepasst:

WW.WORLD_WEATHER_CUR.WEATHER_DESC

WW.WORLD_WEATHER_DAY[0..4].WEATHER_DESC

13.7. WORLD_WEATHER_ICON_OSCAT

Type	Funktionsbaustein:
IN_OUT	WW: WORLD_WEATHER_DATA (Wetterdaten)
INPUT	ACTIVATE: BOOL (positive Flanke startet die Übersetzung)



Der Baustein ersetzt die originalen Anbieterspezifischen Wetter Icons-Nummern durch OSCAT-Standard Icon Nummern. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Icon-Nummern) in der WORLD_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels WORLD_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein WORLD_WEATHER mit ACTIVATE verschalten werden.

Folgende Elemente werden angepasst:

WW.WORLD_WEATHER_CUR.WEATHER_ICON

WW.WORLD_WEATHER_DAY[0..4].WEATHER_ICON

14. Visualisierung

14.1. Wetter-Grafiken

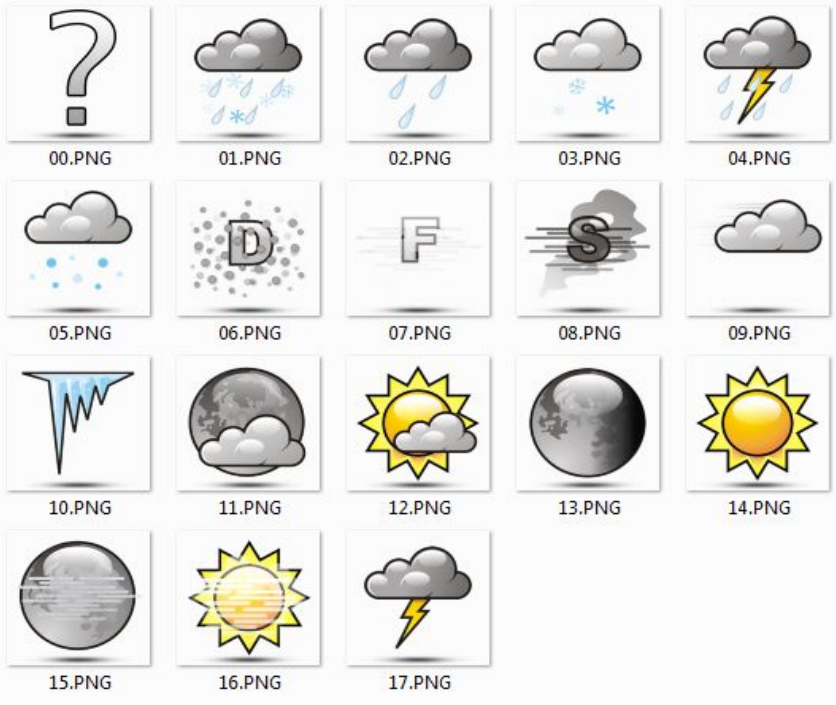
Mit den Wetter-Bausteinen werden die Wetterdaten in den jweiligen Datenstrukturen zur Verfügung gestellt. Standardmäßig liefert jeder Serviceanbieter seine eigenen Wettercode bzw. Wetter-Icons mit. Da sich diese zum Teil total unterscheiden, gibt es für jeden Wetterbaustein eigene Sammlungen.

Mit den Bausteinen

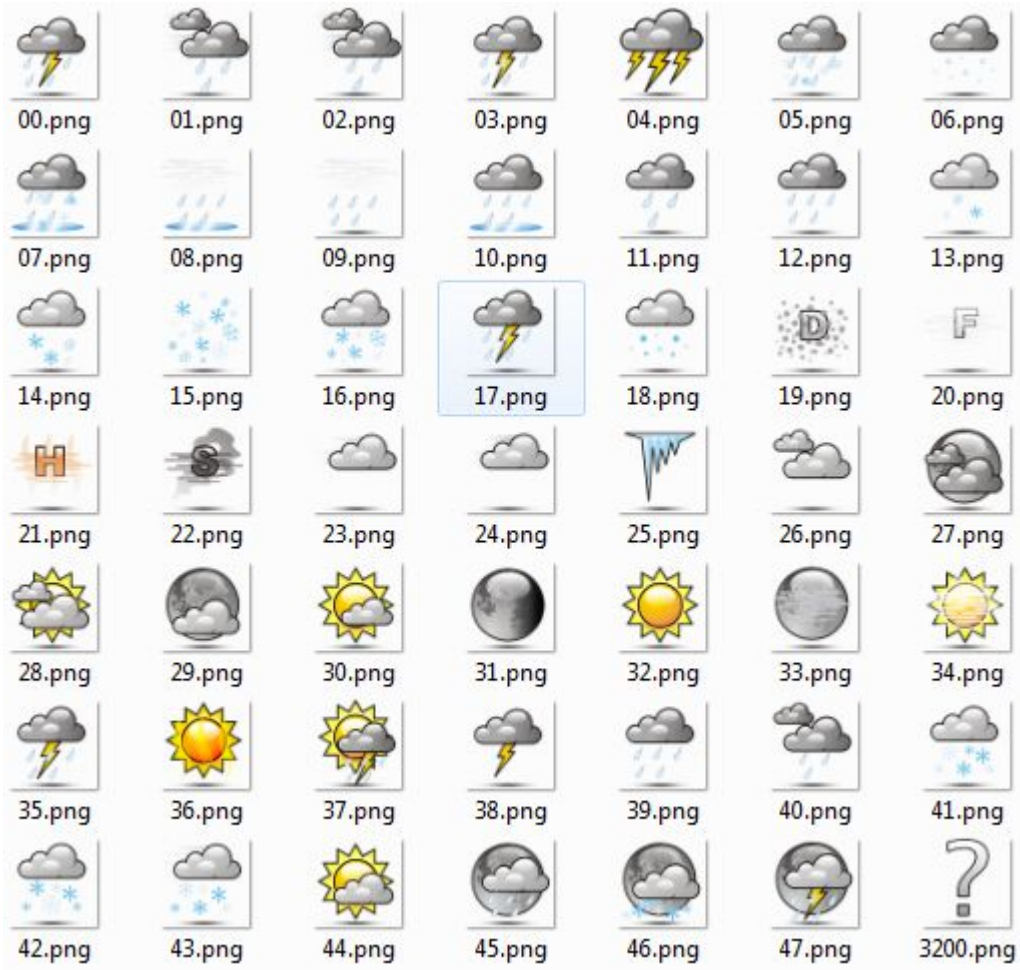
yahoo_weather_icon_oscat.odt
world_weather_icon_oscat.odt

können diese unterschiedlichen Wetterbeschreibungsdaten bzw. Icons auf einen gemeinsamen Nenner (OSCAT-Standard) reduziert werden, sodass ein einziges ICON-Setup ausreicht.

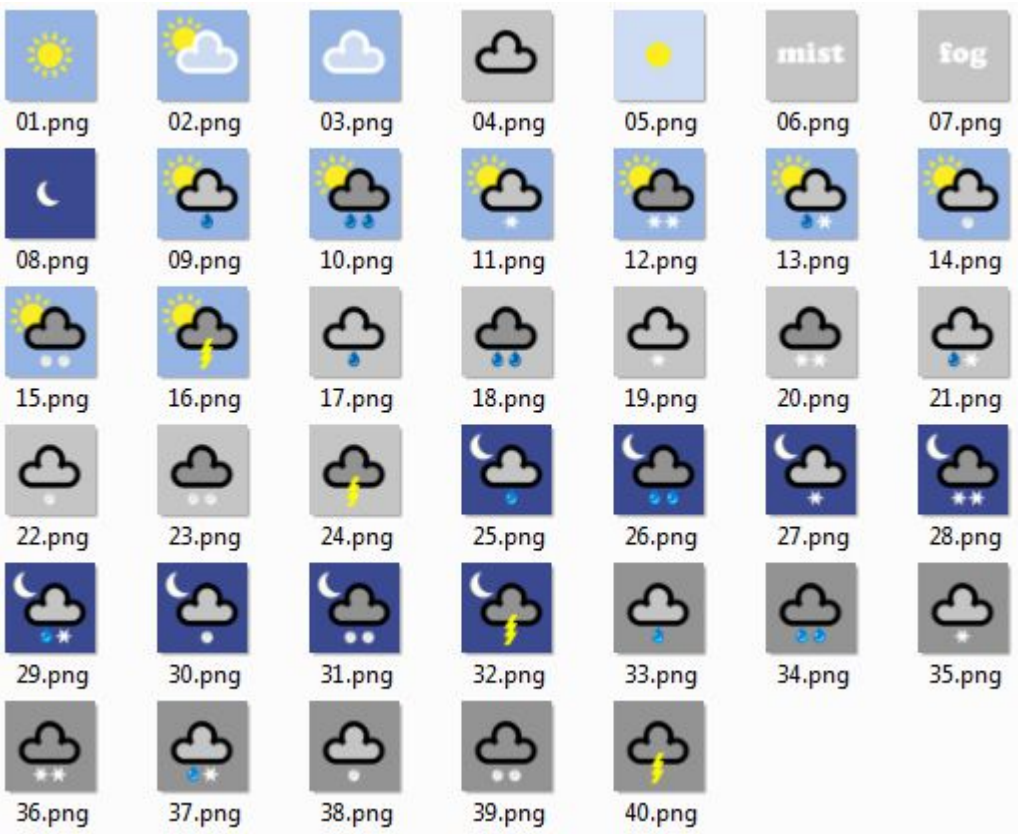
SETUP: **WEATHER_OSCAT_1**



SETUP: **WEATHER_YAHOO_1**

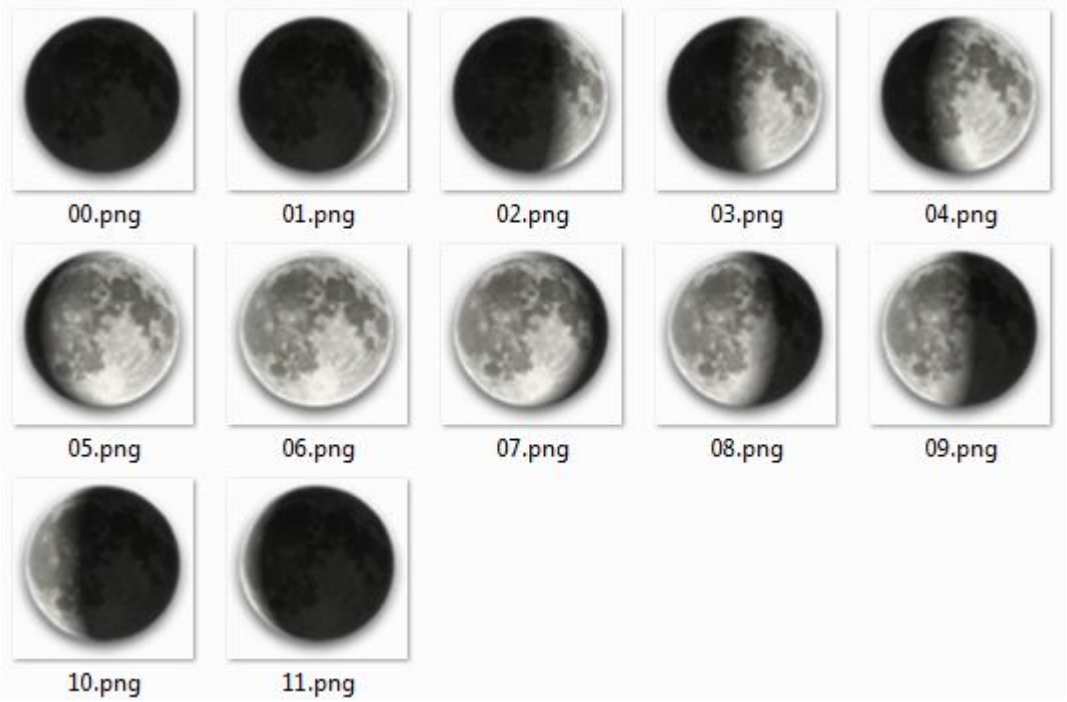


SETUP: **WEATHER_WORLD_1**



14.2. Mond-Grafiken

SETUP: **MOON_1**



14.3. Wind-Grafiken

SETUP: **WIND_1**


n.png
nno.png
no.png
ono.png
o.png
oso.png
so.png
sso.png
s.png
ssw.png
sw.png
wsw.png
w.png
wnw.png
nw.png
nnw.png

Verzeichnis der Funktionsbausteine